

Magy Seif El-Nasr
Anders Drachen
Alessandro Canossa
Editors



Game Analytics

Maximizing the Value
of Player Data

 Springer

Game Analytics

Magy Seif El-Nasr • Anders Drachen
Alessandro Canossa
Editors

Game Analytics

Maximizing the Value of Player Data

 Springer

Editors

Magy Seif El-Nasr
College of Computer and Information Science
College of Arts, Media and Design
Northeastern University
Boston, MA, USA

Anders Drachen
College of Arts, Media and Design
Northeastern University
Boston, MA, USA

Alessandro Canossa
College of Arts, Media and Design
& Center for Computer Games Research
Northeastern University
& IT University of Copenhagen
Boston, MA, USA & Copenhagen, Denmark

Institute of Communication and Psychology
Aalborg University
Copenhagen, Denmark
Game Analytics
Copenhagen, Denmark

Chapter 6 was created within the capacity of an US governmental employment.
US copyright protection does not apply.

Chapter 26 is published with kind permission of Her Majesty the Queen Right of Canada.

ISBN 978-1-4471-4768-8 ISBN 978-1-4471-4769-5 (eBook)
DOI 10.1007/978-1-4471-4769-5
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013933305

© Springer-Verlag London 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover Image: Grete Edland Westerlund

Cover stock images © iStockphoto.com, used with permission

Where stated, images are © Ubisoft Entertainment.

© 2006-2010 Ubisoft Entertainment. All Rights Reserved. Tom Clancy's Splinter Cell, Splinter Cell Double Agent, Sam Fisher, Assassin's Creed, Ubisoft and the Ubisoft logo are trademarks of Ubisoft Entertainment in the U.S. and/or other countries.

Prince of Persia and Prince of Persia The Forgotten Sands are trademarks of Waterwheel Licensing LLC in the US and/or other countries used under license by Ubisoft Entertainment. Based on Prince of Persia® created by Jordan Mechner.

© 2007-2012 Ubisoft Entertainment. All Rights Reserved. Assassin's Creed, Ubisoft and the Ubisoft logo are trademarks of Ubisoft Entertainment in the U.S. and/or other countries.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Over the years, I have spent a fair amount of time teaching game production and design. The most common point of concern has been designers wondering how to gain some degree of self-determination. The greatest points of concern for producers are how to tell if their designer is any good. They usually whisper these questions to me, so the other guy can't hear them.

I tell them the same thing: Designers are in the business of telling the future. Ask them to put their predictions in writing and track how it works out. The results are obvious.

The problem in the past was you could only really track the progress of a designer on a product-by-product basis. That meant measuring them based on each product's success. That isn't really often enough to make much progress as a producer or designer, let alone a game player.

The world had changed. Designers can create new ideas, predict their effect, develop and introduce them to a customer, and measure their results, all in a day. Producers get to see lots of little decisions, and lots of examples of the designers' creativity commercially deployed, for better or worse.

For some designers, this has been scary. That is good. If you can't prove you are right, does it matter? On the other hand, if you can change a product's feature set and improve its financial effectiveness in a repeatable, measurable and meaningful way, won't most Producers leave you alone? After all, they don't know how to do it.

At the end of the day, the truth will set you free. If you can anticipate the behavior of a player and craft that experience to fulfill their expectations, aren't you actually in charge? What game analytics provides, and what this book describes in exhaustive detail, is an understanding that will set you free to concentrate on the parts of the game you can't measure: art – and to make it great. Generally, the numbers we work

in have short return on investment, but the stories and memories we leave behind have the same deep impact that all art has: It changes lives. The numbers are the first tool to get to that opportunity. They unlock the door.

So take the first step and unlock it. Make us believe in you.

42 65 6C 69 65 76 65

Chief Creative Director of Electronic Arts

Rich Hilleman

Acknowledgments

This book took a large amount of time and effort to put together. This involved many people. We would first like to thank the authors who made this book possible: Tim Fields, Sree Santhosh, Mark Vaden, Georg Zoeller, Andre Gagné, Simon McCallum, Jayson Mackie, Christian Thureau, Julian Togelius, Georgios Yannakakis, Christian Bauckhage, Janus Rau Møller Sørensen, Matthias Schubert, Pietro Guardini, Paolo Mannetti, Ben Medler, Dinara Moura, Bardia Aghabeigi, Eric Hazan, Jordan Lynn, Ben Weedon, Veronica Sundstedt, Matthias Bernhard, Efstathios Stavrakis, Erik Reinhard, Michael Wimmer, Lennart Nacke, Graham McAllister, Pejman Mirza-Babaei, Jason Avent, Nicolas Ducheneaut, Nick Yee, Edward Castranova, Travis L. Ross, Isaac Knowles, Jan L. Plass, Bruse D. Homer, Charles K. Kinzer, Yoo Kyung Chang, Jonathan Frye, Walter Kaczetow, Katherine Isbister, Ken Perlin, Carrie Heeter, Yu-Hao Lee, Brian Magerko, and Cameron Brown. All the authors have done more than two revisions of their chapters and have been very open to our consistent nagging for more refinement and changes. Their efforts is what made this book what it is.

We would also like to thank all the people who have allowed us to interview them, sometimes more than once, to revise the information and get more contribution for the book. This includes Jim Baer and Daniel McCaffrey from Zynga, Nicholas Francis and Thomas Hagen from Unity, Darius Kazemi, Aki Järvinen from Digital Chocolate, Nicklas Nygren and Simon Møller from Kiloo, Ola Holmdahl and Ivan Garde from Junebud, and Simon Egenfeldt Nielsen from Serious Games Interactive.

We would also like to thank Alex Kirschner, Brian T. Schnieder, and Bryan Pope from Zynga who have been a fantastic help getting the interview with Jim and Dan scheduled and passing the interview review stage through the communication department. This was a great collaborative effort.

We also thank the people at Game Analytics, notably Christian Thureau and Matthias Flügge, for ongoing feedback on ideas, chapters, and reviews.

We would also like to thank Rich Hilleman for writing the foreword for us. This was a great honor, and Karen Morris for helping set this up and getting everything done on time.

We would also like to thank our reviewers who had to review the chapters, sometimes several chapters and multiple times, to make sure the quality is up to standard. In particular, we thank Bardia Aghabeigi, David Milam, Mark Sivak, Robert Mac Auslan, Mariya Shiyko, Ben Weber, Andre Gagné, Lennart Nacke, Hector Larios, Adam M. Smith, Julian Togelius, Carrie Heeter, Eric Hazan, Georgios Yannakakis, Ian Livingston, Andrea Bonanno, David Tisserand, Ben Medler, Kenneth Hullet, Rasmus Harr, Sofie Mann Harr, Joerg Niesenhaus, Tobias Mahlmann, Christian Thureau, Bill Shribman, Pejman Mirza-Babaei, Tim Marsh, Ben Weedon, Larry Mellon, John Hopson, Brian Meidell, Ben Lile, Bruce Phillips, Andrew Stapleton, Dinara Moura, Tim Ward, Jim Blackhurst, Kristian Kersting, Rafet Sifa, and Heather Desurvire.

We also thank the many companies who kindly permitted their data visualizations, graphs, tables, and other work to be reproduced in the book.

We also thank our respective employers, Northeastern University, Aalborg University, and the IT University of Copenhagen, and GRAND-NCE for funding the cover image for the book.

We are also grateful to Grete Edland Westerlund for her creative input on the cover artwork.

Finally, we direct a heartfelt thanks to our families for their continued and unwavering support throughout the two-year long process of developing the book.

Contents

Part I An Introduction to Game Analytics

1 Introduction	3
Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa	
2 Game Analytics – The Basics	13
Anders Drachen, Magy Seif El-Nasr, and Alessandro Canossa	
3 Benefits of Game Analytics: Stakeholders, Contexts and Domains	41
Alessandro Canossa, Magy Seif El-Nasr, and Anders Drachen	
4 Game Industry Metrics Terminology and Analytics Case Study	53
Timothy Victor Fields	
5 Interview with Jim Baer and Daniel McCaffrey from Zynga	73
Magy Seif El-Nasr and Alessandro Canossa	

Part II Telemetry Collection and Tools

6 Telemetry and Analytics Best Practices and Lessons Learned	85
Sreelata Santhosh and Mark Vaden	
7 Game Development Telemetry in Production	111
Georg Zoeller	
8 Interview with Nicholas Francis and Thomas Hagen from Unity Technologies	137
Alessandro Canossa	
9 Sampling for Game User Research	143
Anders Drachen, André Gagné, and Magy Seif El-Nasr	

10	WebTics: A Web Based Telemetry and Metrics System for Small and Medium Games	169
	Simon McCallum and Jayson Mackie	
11	Interview with Darius Kazemi	195
	Magy Seif El-Nasr	
Part III Game Data Analysis		
12	Game Data Mining	205
	Anders Drachen, Christian Thureau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage	
13	Meaning in Gameplay: Filtering Variables, Defining Metrics, Extracting Features and Creating Models for Gameplay Analysis	255
	Alessandro Canossa	
14	Gameplay Metrics in Game User Research: Examples from the Trenches	285
	Anders Drachen, Alessandro Canossa, and Janus Rau Møller Sørensen	
15	Interview with Aki Järvinen from Digital Chocolate	321
	Alessandro Canossa	
16	Better Game Experience Through Game Metrics: A Rally Videogame Case Study	325
	Pietro Guardini and Paolo Maninetti	
Part IV Metrics Visualization		
17	Spatial Game Analytics	365
	Anders Drachen and Matthias Schubert	
18	Visual Game Analytics	403
	Ben Medler	
19	Visual Analytics Tools – A Lens into Player’s Temporal Progression and Behavior	435
	Magy Seif El-Nasr, André Gagné, Dinara Moura, and Bardia Aghabeigi	
20	Interview with Nicklas “Niffas” Nygren	471
	Alessandro Canossa	
Part V Mixed Methods for Game Evaluation		
21	Contextualizing Data	477
	Eric Hazan	

22 Combining Back-End Telemetry Data with Established User Testing Protocols: A Love Story 497
Jordan Lynn

23 Game Metrics Through Questionnaires 515
Ben Weedon

24 Interview with Simon Møller from Kiloo 539
Alessandro Canossa

25 Visual Attention and Gaze Behavior in Games: An Object-Based Approach 543
Veronica Sundstedt, Matthias Bernhard, Efstathios Stavarakis, Erik Reinhard, and Michael Wimmer

26 An Introduction to Physiological Player Metrics for Evaluating Games 585
Lennart E. Nacke

27 Improving Gameplay with Game Metrics and Player Metrics 621
Graham McAllister, Pejman Mirza-Babaei, and Jason Avent

Part VI Analytics and Player Communities

28 Data Collection in Massively Multiplayer Online Games: Methods, Analytic Obstacles, and Case Studies..... 641
Nicolas Ducheneaut and Nick Yee

29 Designer, Analyst, Tinker: How Game Analytics Will Contribute to Science 665
Edward Castronova, Travis L. Ross, and Isaac Knowles

30 Interview with Ola Holmdahl and Ivan Garde from Junebud 689
Alessandro Canossa

Part VII Metrics and Learning

31 Metrics in Simulations and Games for Learning 697
Jan L. Plass, Bruce D. Homer, Charles K. Kinzer, Yoo Kyung Chang, Jonathan Frye, Walter Kaczetow, Katherine Isbister, and Ken Perlin

32 Conceptually Meaningful Metrics: Inferring Optimal Challenge and Mindset from Gameplay 731
Carrie Heeter, Yu-Hao Lee, Ben Medler, and Brian Magerko

**33 Interview with Simon Egenfeldt Nielsen
from Serious Games Interactive 763**
Alessandro Canossa

Part VIII Metrics and Content Generation

34 Metrics for Better Puzzles 769
Cameron Browne

Erratum E1

Contributors

Bardia Aghabeigi Northeastern University, Boston, MA, USA

Jason Avent Disney Interactive Studios, Glendale, CA, USA

Christian Bauckhage Fraunhofer IAIS and the University of Bonn, Bonn, Germany

Matthias Bernhard Vienna University of Technology, Vienna, Austria

Cameron Browne Imperial College London, London, UK

Alessandro Canossa College of Arts, Media and Design, Northeastern University, Boston, MA, USA

Center for Computer Games Research, IT University, Copenhagen, Denmark

Edward Castronova Department of Telecommunications, Indiana University, Bloomington, IN, USA

Yoo Kyung Chang Games for Learning Institute (G4LI), Teachers College Columbia University, New York, NY, USA

Anders Drachen PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark

Nicolas Ducheneaut Palo Alto Research Center, Palo Alto, CA, USA

Jonathan Frye Games for Learning Institute (G4LI), New York University, New York, NY, USA

André Gagné THQ, Agoura Hills, CA, USA

Pietro Guardini Milestone S.r.l, Milan, Italy

Eric Hazan Ubisoft, Montreal, France

Carrie Heeter Department of Telecommunication, Information Studies, and Media, Michigan State University, Lansing, MI, USA

Bruce D. Homer Games for Learning Institute (G4LI), CUNY Graduate Center, New York, NY, USA

Katherine Isbister Games for Learning Institute (G4LI), New York University, New York, NY, USA

Walter Kaczetow Games for Learning Institute (G4LI), CUNY Graduate Center, New York, NY, USA

Charles K. Kinzer Games for Learning Institute (G4LI), Teachers College Columbia University, New York, NY, USA

Isaac Knowles Department of Telecommunications, Indiana University, Bloomington, IN, USA

Yu-Hao Lee Media & Information Studies, Michigan State University, Lansing, MI, USA

Jordan Lynn Volition, Champaign, IL, USA

Jayson Mackie Gjøvik University College, Gjøvik, Norway

Brian Magerko Digital Media in the School of Literature, Communication, and Culture, Georgia Institute of Technology, Atlanta, GA, USA

Paolo Maninetti Milestone S.r.l, Milan, Italy

PopCap Games International, The Academy, Dublin 2, Ireland

Graham McAllister Player Research, Hove, East Sussex, UK

Simon McCallum Gjøvik University College, Gjøvik, Norway

Ben Medler Georgia Tech University, Atlanta, GA, USA Georgia Institute of Technology, Atlanta, GA, USA

Pejman Mirza-Babaei University of Sussex, Brighton, UK

Dinara Moura School of Interactive Arts and Technology, Simon Fraser University, Surrey, BC, Canada

Lennart E. Nacke HCI and Game Science Group, Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON, Canada

Ken Perlin Games for Learning Institute (G4LI), New York University, New York, NY, USA

Jan L. Plass Games for Learning Institute (G4LI), New York University, New York, NY, USA

Erik Reinhard Max Planck Institute for Informatics, Saarbrücken, Germany

Travis L. Ross Department of Telecommunications, Indiana University, Bloomington, IN, USA

Sreelata Santhosh Online Technology Group, Sony Computer Entertainment America, San Diego, CA, USA

Matthias Schubert Institute for Informatics, Ludwig-Maximilians-Universität, Munich, Germany

Magy Seif El-Nasr PLAIT Lab, College of Computer and Information Science, College of Arts, Media and Design, Northeastern University, Boston, MA, USA

College of Computer and Information Science, Northeastern University, Boston, MA, USA

Janus Rau Møller Sørensen Crystal Dynamics/Square Enix, Redwood City, CA, USA

Efstathios Stavrakis Department of Computer Science, University of Cyprus, Lefkosia, Cyprus

Veronica Sundstedt Blekinge Institute of Technology, Karlskrona, Sweden

Christian Thureau Game Analytics, Ballerup, Denmark

Julian Togelius Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark

Mark Vaden Online Technology Group, Sony Computer Entertainment America, San Diego, CA, USA

Timothy Victor Fields Capcom, Chuo-ku, Osaka, Japan

Ben Weedon PlayableGames, London, UK

Michael Wimmer Institute for Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria

Georgios N. Yannakakis Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark

Nick Yee Palo Alto Research Center, Palo Alto, CA, USA

Georg Zoeller Ubisoft Singapore, Singapore

Part I

An Introduction to Game Analytics

Game analytics is not an altogether new or independent field. It has roots in and borrows largely from many existing fields, such as usability inspection methods, business intelligence, statistics and data mining, amongst others. It is therefore necessary to provide a panoramic view on the key disciplines and concepts that are at the core of game analytics.

This part has the following take-aways:

- Show the recent history of game analytics and introduce this fascinating area
- Introduce key concepts and disciplines
- Discuss benefits for the different stakeholders in the game industry
- Define reoccurring terms and concepts used as measures in social games.

The part will consist of five chapters:

- Chapter 1 provides an introduction of the book outlining the different parts and the chapters of the book.
- Chapter 2 provides the basics of telemetry, definitions, uses, and concepts.
- Chapter 3 outlines the benefits of the telemetry data and analytics to the different stakeholders within the industry.
- Chapter 4: *Game Industry Metrics Terminology and Analytics Case Study* is a contribution from Tim Fields, a veteran producer, game designer, team leader and business developer, who has been building games professionally since 1994. In his chapter, Tim introduces the terminologies used within the social game industry to outline major metrics used currently within the industry with a case study to supplement the discussion.
- Chapter 5: *Interview with Jim Baer and Daniel McCaffrey from Zynga* is an interview with Jim Baer, Senior Director of Analytics, and Daneil McCaffery, Senior Director of Platform and Analytics Engineering, from Zynga outlining Zynga's use of game analytics and their view and future as they expand on this field.

Chapter 1

Introduction

Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa

1.1 Changing the Game

Game Analytics has gained a tremendous amount of attention in game development and game research in recent years. The widespread adoption of data-driven business intelligence practices at operational, tactical and strategic levels in the game industry, combined with the integration of quantitative measures in user-oriented game research, has caused a paradigm shift. Historically, game development has not been data-driven, but this is changing as the benefits of adopting and adapting analytics to inform decision making across all levels of the industry are becoming generally known and accepted.

M. Seif El-Nasr, Ph.D. (✉)
PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

A. Drachen, Ph.D.
PLAIT Lab, Northeastern University, Boston, MA, USA
Department of Communication and Psychology, Aalborg University, Aalborg, Denmark
Game Analytics, Copenhagen, Denmark
e-mail: andersdrachen@gmail.com

A. Canossa, Ph.D.
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

While analytics practices play a role across all aspects of a company, the introduction of analytics in game development has, to a significant extent, been driven by the need to gain better knowledge about the users – the players. This need has been emphasized with the rapid emergence of social online games and the Free-to-Play business model which, heavily inspired by web- and mobile analytics, relies on analysis of comprehensive user behavior data to drive revenue. Outside of the online game sector, users have become steadily more deeply integrated into the development process thanks to widespread adoption of user research methods. Where testing used to be all about browbeating friends and colleagues into finding bugs, user testing and research today relies on sophisticated methods to provide feedback directly on the design.

Operating in the background of these effects is the steady increase in the size of the target audience for games, as well as its increasing diversification. This has brought an opportunity for the industry to innovate on different forms of play allowing different types of interactions and contexts, and the accommodation of different types of users of all ages, intellectual abilities, and motivations. Now, more than ever, it is necessary for designers to develop an understanding of the users and the experiences they obtain from interacting with games. This has marked the birth of Games User Research (GUR) – a still emerging field but an important area of investment and development for the game industry, and one of the primary drivers in establishing analytics as a key resource in game development.

Game analytics is, thus, becoming an increasingly important area of business intelligence for the industry. Quantitative data obtained via telemetry, market reports, QA systems, benchmark tests, and numerous other sources all feed into business intelligence management, informing decision-making. Measures of processes, performance and not the least user behaviors collected and analyzed over the complete life cycle of a game – from cradle to grave – provides stakeholders with detailed information on every aspect of their business. From detailed feedback on design, snapshots of player experience, production performance and the state of the market. Focusing on user-focused analytics, there are multiple uses in the development pipeline, including the tracking and elimination of software bugs, user preferences, design issues, behavior anomalies, and monetization data, to mention a few.

1.2 About This Book

This book is about *game analytics*. It is meant for anybody to pick up – novice or expert, professional or researcher. The book has content for everyone interested in game analytics.

The book covers a wide range of topics under the game analytics umbrella, but has a running focus on the *users*. Not only is ‘user-oriented analytics’ one of the main drivers in the development of game analytics, but users are, after all, the people games are made for. Additionally, the contributions in this book – written by experts in their respective domains – focus on *telemetry* as a data source for analytics.

While not the only source of game business intelligence, telemetry is one of the most important ones when it comes to user-oriented analytics, and has in the past decade brought unprecedented power to Game User Research.

The book is composed of chapters authored by professionals in the industry as well as researchers, and in several cases in collaboration. These are augmented with a string of interviews with industry experts and top researchers in game analytics. This brings together the strengths of both worlds (the industry and academic) and provides a book with a broad selection of in-depth examples of the application of user-oriented game analytics. It also means the book presents a coherent picture of how game analytics can be used to analyze user behavior in the service of stakeholders in both the industry and academia, including: designers who want to know how to change games for building ultimate experiences and boosting retention, business VPs hoping to increase their product sales, psychologists interested in understanding human behavior, computer scientists working on data mining of complex datasets, learning scientists who are interested in developing games that are effective learning tools, game user research methodologists who are interested in developing valid methods to tackle the question of game user experience measurement and evaluation.

Chapters in this book provide a wealth of experiences and knowledge; the urging purpose behind the book is to share knowledge and experiences of the pros and cons of various techniques and strategies in game analytics – including different collection, analysis, visualization and reporting techniques – the building blocks of game analytics systems. In addition, the book also serves to inform practitioners and researchers of the variety of uses and the value of analytics across the game lifecycle, and about the current open problems. It is our ultimate goal to stimulate the existing relations between industry and research, and take the first step towards building a methodological and theoretical foundation for game analytics.

1.3 Game Analytics, Metrics and Telemetry: What Are They?

In this book you will see the following words often repeated: *game metrics*, *game telemetry* and *game analytics*. These terms are today often used interchangeably, primarily due the relative recent adoption of the terms analytics, telemetry and metrics in game development. To clear away any confusion, let us quickly define them. *Game analytics* is the application of analytics to game development and research. The goal of game analytics is to support decision making, at operational, tactical and strategic levels and within all levels of an organization – design, art, programming, marketing, user research, etc. Game analytics forms a key source of business intelligence in game development, and considers both games as products, and the business of developing and maintaining these products. In recent years, many game companies – from indie to AAA – have started to collect *game telemetry*. Telemetry is data obtained over a distance. This can, for example, be quantitative data about how a user plays a game, tracked from the game client and transmitted to

a collection server. *Game metrics* are interpretable measures of something related to games. More specifically, they are quantitative measures of attributes of objects. A common source of game metrics is telemetry data of player behavior. This raw data can be transformed into metrics, such as “total playtime” or “daily active users” – i.e. measures that describe an attribute or property of the players. Metrics are more than just measures of player behavior, however; the term covers any source of business intelligence that operates in the context of games. Chapter 2 delves deeper to outline the definitions of the terms and concepts used within the different chapters in the book.

1.4 User-Oriented Game Analytics

The game industry is inherently diverse. Companies have established their own processes for game analytics, which tend to be both similar and different across companies, depending on the chosen business model, core design features and the intended target audience.

To start with the sector of the industry that relies directly and heavily on user-oriented analytics, *social online game* companies produce games that are played within a social context, either synchronously or asynchronously between a small or large number of players over a server. Many games supporting large-scale multi-player interaction feature a persistent world that users interact within. For these types of games, and social online games in general, companies can release patches at any time and most of the time they add or adjust the game during the lifecycle of the product. Due to this flexibility, companies that produce these types of games usually release the product early and then utilize massive amounts of game telemetry analysis to adjust the game and release new content based on what players are doing. Companies that produce these types of games include Zynga Inc. and Blizzard Entertainment, to mention a few. Chapter 4 delves a bit deeper on the process involved in creating these types of games.

In addition to social game companies, the traditional one-shot retail game model comprises the majority of the industry, today. In this category we find the big franchises like Assassin’s Creed (Ubisoft), Tomb Raider (Square Enix) and NBA (Electronic Arts). Most of the time these games do not feature persistent worlds, and thus do not have the same degree of opportunity to adjust products after launch on a running basis, although this may be changing due to the presence of online distribution networks like Valve’s Steam. However, during production, user-oriented analytics can be used for a large variety of purposes, not the least to help user research departments assist designers in between iterations. This book includes multiple examples of this kind of analytics work, including Eric Hazan’s chapter (Chap. 21) describing the methodologies used at Ubisoft to measure the user experience, Drachen et al. (Chap. 14) describing user research at Crystal Dynamics and IO Interactive, and Jordan Lynn’s chapter (Chap. 22) describing the methods of value to Volition, Inc. Another interesting example of the use of analytics within

the production cycle at Bioware is discussed at length by Georg Zoeller (Chap. 7). Sree Santhosh and Mark Vaden describe their work at Sony Online Entertainment (Chap. 6) and Tim Fields provides an overview of metrics for social online games (Chap. 4).

Of course, recently there has been a mix of AAA titles that also have social or casual components played online. These include Electronic Arts (EA) Sport's FIFA game, which includes an online component with a persistent world. For these games, a mix of approaches and processes are applicable.

1.5 User-Oriented Game Research

As discussed above, Game User Research is a field that studies user behavior. The field is dependent on the methodologies that have been developed in academia, such as quantitative and qualitative methods used within human-computer interaction, social science, psychology, communications and media studies. Digital games present an interesting challenge as they are interactive, computational systems, where engagement is an important factor. For such systems, academics within the user research area have been working hard to adopt and extend the methodologies from other fields to develop appropriate tools for games.

Looking at game analytics specifically, industry professionals and researchers have collaborated to push the frontier for game analytics and analytics tools. Some of this work is covered in Drachen et al.'s work on spatial analysis (Chap. 17) and game data mining (Chap. 12), showing examples of analysis work in games developed and published by Square Enix studios. Also, Medler's work with Electronic Arts (Chap. 18) where he explored the use of different visualization techniques to serve different stakeholders, and Seif El-Nasr et al.'s work with Pixel Ante and Electronic Arts (Chap. 19) where they explored the development of novel visual analytics systems that allow designers to make sense of spatial and temporal behavioral data.

Researchers in the game user research area have been pushing the frontier of methods and techniques in several directions. Some researchers have started to explore triangulation of data from several sources, including metrics and analytics with other qualitative techniques. Examples of these innovative methodologies can be seen in this book. For example, Sundstedt et al.'s chapter (Chap. 25) discusses eye tracking metrics as a behavioral data source, and McAllister et al.'s chapter (Chap. 27), which follows Nacke's chapter (Chap. 26) introduction to physiological measures with a presentation of a novel method triangulating game telemetry with physiological measures.

In addition to innovation in tools and techniques that can be used in industry and research, experts in social sciences, communication, and media studies have also been exploring the use of analytics to further our understanding of human behavior within virtual environments, and, thus, producing insights for game design. In addition, the utility of games for learning has been explored. Examples of this work are included

in the chapters by Ducheneaut and Yee (Chap. 28), Castranova et al. (Chap. 29), Heeter et al. (Chap. 32) and Plass et al. (Chap. 31).

1.6 Structure of This Book

The book is divided into several parts, each highlighting a particular aspect of game analytics for development and research, as follows:

Part I: An Introduction to Game Analytics introduces the book, its aims and structure. This part will contain four chapters. The first chapter (*Introduction*), which you are reading now, is a general introduction of the book outlining the different parts and chapters of the book. Chapter 2 (*Game Analytics – The Basics*) forms the foundation for the book’s chapters, outlining the basics of game analytics, introducing key terminology, outlines fundamental considerations on attribute selection and the role of analytics in game development and the knowledge discovery process. Chapter 3 (*The Benefits of Game Analytics: Stakeholders, Contexts and Domains*) discusses the benefits of metrics and analytics to the different stakeholders in industry and research. Chapter 4 (*Game Industry Metrics Terminology and Analytics Case Study*) is a contribution from Tim Fields, a veteran producer, game designer, team leader and business developer, who has been building games professionally since 1994. In his chapter, Tim introduces the terminologies used within the social game industry to outline major metrics used currently within the industry with a case study to supplement the discussion. Chapter 5 (*Interview with Jim Baer and Daniel McCaffrey from Zynga*) is an interview with Zynga – a company that has been on the forefront of game analytics and its use within social games as an important process to push the business and design of games. This chapter will outline their use of game analytics, the systems they developed and their view of the fields’ future.

Part II: Telemetry Collection and Analytics Tools is composed of six chapters, and describes methods for telemetry collection and tools used within the industry for that purpose. In particular, we have five chapters in this part of the book. Chapter 6 (*Telemetry and Analytics Best Practices and Lessons Learned*) is a contribution from Sony Entertainment discussing a tool they have developed and used within the company for several years to collect and analyze telemetry data within Sony’s pipeline. The chapter outlines best practices after iterating over this system for years. Chapter 7 (*Game Development Telemetry in Production*) is another industry chapter contributed by Georg Zoeller. In this chapter, he discusses a game analytics system he developed to enable the company to collect and analyze game metrics during production to specifically aid in workflow, quality assurance, bug tracking, and pre-launch design issues. Chapter 8 follows by an interview (*Interview with Nicholas Francis and Thomas Hagen from Unity*) outlining Unity Technologies’ view of tool development within the Unity 3D platform for telemetry collection and analysis. In addition to how to collect game telemetry, who to collect this data from is of equal

importance. Chapter 9 (*Sampling for Game User Research*) addresses this issue by discussing best practices in sampling, borrowing from social science research and how to best apply such sampling techniques to game development. This chapter is a contribution from Anders Drachen and Magy Seif El-Nasr in collaboration with Andre Gagné, a user researcher at THQ. Next, Chap. 10 (*WebTics: A Web Based Telemetry and Metrics System for Small and Medium Games*) describes an open source middleware tool under development intended for small-medium scale developers, and discusses telemetry collection from a practical standpoint. This chapter is a contribution from Simon McCallum and Jayson Mackie, both researchers at Gjøvik University College, Norway. The part closes with a Chap. 11 (*Interview with Darius Kazemi*), an interview with Darius Kazemi, a game analytics veteran with over 10 years of experience analyzing game telemetry from games as diverse as casual and AAA titles. The interview focuses on game analytics in general, the current state of the industry and what he sees as the future for analytics in game development.

Part III Game Data Analysis, composed of five chapters, addresses analysis methods for the data collected. Specifically, it introduces the subject of datamining as an analysis method: Chapter 12 (*Game Data Mining*), a contribution from Anders Drachen and Christian Thureau, CTO of Game Analytics, a middleware company delivering game analytics services to the industry, Julian Togelius, Associate professor at The IT University Copenhagen, Georgios Yannakakis, Associate professor at University of Malta, and Christian Bauckhage, professor at the University of Bonn, Germany. The part will also discuss data collection, metrics, telemetry and abstraction of this data to model behavior, which is the subject of Chap. 13 (*Meaning in Gameplay: Filtering Variables, Defining Metrics, Extracting Features and Creating Models for Gameplay Analysis*), a contribution from Alessandro Canossa. Additionally, this part will also include case studies to show analysis in action: Chapter 14 (*Gameplay Metrics in Game User Research: Examples from the Trenches*), a contribution from Anders Drachen and Alessandro Canossa with Janus Rau Møller Sørensen, a user research manager at Crystal Dynamics and IO Interactive, worked on titles including Hitman Absolution, Tomb Raider and Deus Ex: Human Revolution, and Chap. 16 (*Better Game Experience through Game Metrics: A Rally Videogame Case Study*), a contribution from Pietro Guardini, games user researcher at Milestone, who has contributed to several titles, including *MotoGP 08* and the *Superbike World Championship* (SBK), and Paolo Maninetti, senior game programmer at Milestone, who has worked on titles such as *MotoGP 08* and the *Superbike World Championship* (SBK). This part of the book also includes an interview with Aki Järvinen, creative director and competence manager at Digital Chocolate (Chap. 15: *Interview with Aki Järvinen from Digital Chocolate*), discussing the use of analytics at Digital Chocolate and its role and importance within the company.

Part IV: Metrics Visualization deals with visualization methods of game metrics as a way of analyzing data or showing the data to stakeholders. This part has four chapters. The part starts with an introduction to the area of spatial and temporal game

analytics which is the subject of Chap. 17 (*Spatial Game Analytics*). The chapter is a contribution from Anders Drachen with Matthias Shubert who is a professor at Ludwig-Maximilians-Universität. The following two chapters delve deeper into case studies with visualization tools for game telemetry analysis. In particular, Chap. 18 (*Visual Game Analytics*) discusses visual analytics tools developed for Electronic Arts' *Dead Space* team, a contribution from Ben Medler, a PhD student at Georgia Tech who worked in collaboration with Electronic Arts as a graduate researcher. Chapter 19 (*Visual Analytics tools – A Lens into Player's Temporal Progression and Behavior*) a contribution from Magy Seif El-Nasr, Andre Gagné, a user researcher at THQ, Dinara Moura, PhD student at Simon Fraser University, Bardia Aghabeigi, PhD student at Northeastern University and a game analytics researcher at Blackbird Interactive. The chapter discusses two case studies of visual analytics tools developed for two different games and companies: an RTS game developed by Pixel Ante as a free to play single player game and an RPG game developed by Bioware. The part concludes with Chap. 20 (*Interview with Nicklas "Nifflas" Nygren*) an interview with an independent game developer working in Sweden and Denmark, that introduces his views, as an indie developer, on game analytics.

Part V: Mixed Methods for Game Evaluation, consists of seven chapters addressing multiple methods used for game evaluation. These methods include triangulation techniques for telemetry and qualitative data – subject of Chap. 21 (*Contextualizing Data*) with case studies from Eric Hazan, a veteran user researcher at Ubisoft and Chap. 22 (*Combining Back-End Telemetry Data with Established User Testing Protocols: A Love Story*) with case studies from Jordan Lynn a veteran user researcher at Volition, Inc. In addition to triangulation methods, this part also features the use of metrics extracted from surveys as discussed in Chap. 23 (*Game Metrics Through Questionnaires*), a contribution from Ben Weedon, consultant and manager at PlayableGames, a games user research agency in London, UK. Chapter 25 (*Visual Attention and Gaze Behavior in Games: An Object-Based Approach*) discusses the use of eye tracking as metrics for game evaluation, a contribution from Veronica Sundstedt, lecturer at Blekinge Institute of Technology, Matthias Bernhard, PhD candidate at Vienna University of Technology, Efstathios Stavrakis, researcher at University of Cyprus, Erik Reinhard, researcher at Max Plank Institute of Informatics, and Michael Wimmer, professor at Vienna University of Technology. Chapter 26 (*An Introduction to Physiological Player Metrics for Evaluating Games*), a contribution from Lennart Nacke, assistant professor at University of Ontario Institute of Technology, and Chap. 27 (*Improving Gameplay with Game Metrics and Player Metrics*), a contribution from Graham McAllister, director of Vertical Slice, a game user research company, Pejman Mirza-Babaei, PhD candidate at the University of Sussex, and Jason Avent, Disney Interactive Studios, both investigate the use of psycho-physiological metrics for game evaluation. The part also includes an interview with Simon Møller Chap. 24 (*Interview with Simon Møller from Kiloo*) creative director at Kiloo, a publisher and independent development company pushing a new model for co-productions. The chapter explores' the founders perspective on game analytics for mobile development.

Part VI: Analytics and Player Communities discusses case studies for understanding social behavior of player communities. Chapter 28 (*Data Collection in Massively Multiplayer Online Games: Methods, Analytic Obstacles, and Case Studies*) is a contribution by Nic Ducheneaut, senior scientist, and Nick Yee, research scientist, both at PARC. Chapter 29 focuses on general design perspectives (*Designer, Analyst, Tinker: How Game Analytics will Contribute to Science*), a contribution by Edward Castronova, Travis L. Ross and Issac Knowles, researchers from Indiana University. This part also includes an interview Chap. 30 (*Interview with Ola Holmdahl and Ivan Garde from Junebud*) with Ola Holmdahl, the founder and CEO of Junebud and Ivan Garde, producer, business and metrics analyst, also at Junebud. This interview explores the use of metrics for web-based MMOGs.

Part VII: Metrics and Learning includes two chapters that focus on metrics for pedagogical evaluation. These are Chaps. 31 and 32: Chapter 31 (*Metrics in Simulations and Games for Learning*) and Chap. 32 (*Conceptually Meaningful Metrics: Inferring Optimal Challenge and Mindset from Gameplay*). The former is a contribution from Jan Plass, Games for Learning Institute, New York Polytechnic, in collaboration with Bruce D. Homer and Walter Kaczetow from the City University of New York (CUNY) Graduate Center; Charles K. Kinzer and Yoo Kyung Chang from Teachers College Columbia University, and Jonathan Frye, Katherine Isbister and Ken Perlin from New York University. Chapter 32 is a contribution from by Carrie Heeter, professor at Michigan State University and Yu-Hao Lee, PhD student from Michigan State University, with Ben Medler (see title above) and Brian Magerko, assistant professor at Georgia Tech University. In addition to these two chapters, this part of the book features an interview with Simon Egenfeldt Nielsen, CEO of Serious Games Interactive, exploring the use of analytics for serious games from an industry perspective in Chap. 33 (*Interview with Simon Egenfeldt Nielsen from Serious Games Interactive*).

Part VIII: Metrics and Content Generation, discusses the emerging application of game metrics in procedural content generation. Chapter 34 (*Metrics for Better Puzzles*), by Cameron Browne from the Imperial College London, builds a case for using metrics to generate content in puzzle games.

About the Editors

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in

several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. *Magy worked collaboratively with Electronic Arts, Bardel Entertainment, and Pixel Ante.*

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and the Royal Danish Academy of Fine Arts, Schools of Architecture, Design and Conservation. His doctoral research was carried out in collaboration with IO Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches. His work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he maintains an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio.

Chapter 2

Game Analytics – The Basics

Anders Drachen, Magy Seif El-Nasr, and Alessandro Canossa

Take Away Points:

- Overview of important key terms in game analytics.
- Introduction to game telemetry as a source of business intelligence.
- In-depth description and discussion of user-derived telemetry and metrics.
- Introduction to feature selection in game analytics.
- Introduction to the knowledge discovery process in game analytics.
- References to essential further reading.

A. Drachen, Ph.D. (✉)

PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark

e-mail: andersdrachen@gmail.com

M. Seif El-Nasr, Ph.D.

PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

A. Canossa, Ph.D.

College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark

e-mail: a.canossa@neu.edu

2.1 Analytics – A New Industry Paradigm

Developing a profitable game in today's market is a challenging endeavor. Thousands of commercial titles are published yearly, across a number of hardware platforms and distribution channels, all competing for players' time and attention, and the game industry is decidedly competitive. In order to effectively develop games, a variety of tools and techniques from e.g. business practices, project management to user testing have been developed in the game industry, or adopted and adapted from other IT sectors. One of these methods is *analytics*, which in recent years has decidedly impacted on the game industry and game research environment.

Analytics is the process of discovering and communicating patterns in data, towards solving problems in business or conversely predictions for supporting enterprise decision management, driving action and/or improving performance. The methodological foundations for analytics are statistics, data mining, mathematics, programming and operations research, as well as data visualization in order to communicate insights learned to the relevant stakeholders. Analytics is not just the querying and reporting of BI (Business Intelligence) data, but rests on actual analysis, e.g. statistical analysis, predictive modeling, optimization, forecasting, etc. (Davenport and Harris 2007).

Analytics typically relies on computational modeling. There are several branches or domains of analytics, e.g. marketing analytics, risk analytics, web analytics – and game analytics. Importantly, analytics is not the same thing as data analysis. Analytics is an umbrella term, covering the entire methodology of finding and communicating patterns in data, whereas analysis is used for individual applied instances, e.g. running a particular analysis on a dataset (Han et al. 2011; Davenport and Harris 2007; Jansen 2009).

Analytics forms an important subset of, and source of, **Business Intelligence** (BI) across all levels of a company or organization, irrespective of its size. BI is a broad concept, but basically the goal of BI is to turn raw data into useful information. BI refers to any method (usually computer-based) for identifying, registering, extracting and analyzing business data, whether for strategic or operational purposes (Watson and Wixom 2007; Rud 2009). Common for all business intelligence is the aim to provide support for decision-making at all levels of an organization – as defined by Luhn (1958): “the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal.” In essence, the goal of BI – and by extension game analytics – is to provide a means for a company to become data-driven in its strategies and practices.

In the context of the ICT industry, BI covers a variety of data sources from the *market* (benchmark reports, white papers, market reports), the *company* in question (QA reports, production updates, budgets and business plans) and not the least the *users* (players, customers) of the company's games (user test reports, user research, customer support analysis). These sources of BI operate across temporal (historical as well as predictive) and geographical distances as well as across products. **Game analytics** is a specific application domain of analytics, describing it as applied in the

context of game development and game research. The direct benefit gained from adopting game analytics is support for decision-making at all levels and all areas of an organization – from design to art, programming to marketing, management to user research. Game analytics is directed at both the analysis of the *game as a product*, e.g. whether it provides a good user experience (Law et al. 2007; Nacke and Drachen 2011) and the *game as a project*, e.g. the process of developing the game, including comparison with other games (benchmarking).

Just like “regular” analytics in the IT sector in general, game analytics is concerned with all forms of data that pertains to game business or research – not just data about user behavior or from user testing. This is a common misconception because the analysis of user behavior has been an important driver for the evolution of game analytics in the past decade, and because in the cousin fields: web analytics and mobile analytics – two of the strongest sources of inspiration for game analytics – customer behavior analysis is a key area. Game analytics is a young domain, where there has yet to emerge a standard set of key terms and processes. Such standards exist in other sub-domains of analytics, e.g. web analytics, providing models for establishing such frameworks in game analytics in the future (WAA 2007).

To sum up, game analytics is business analytics adapted to the specific context of games. This by extension makes the domain of game analytics fairly broad and too cumbersome a topic to be treated in detail in any one book. Indeed, business intelligence, analytics, big data, data-driven business practices and related topics are the subject of numerous books, white papers, reports and research articles, and it is not possible in this chapter – nor this book – to provide a foundation for the entire field of game analytics. In this chapter a brief introduction is provided focusing on the topics that the chapters in this book focus on: while this book covers a range of topics on game analytics, the chapters are generally – but not exclusively – focused on two aspects of game analytics:

1. **Telemetry:** The chapters in this book focus on a particular source of data used in game analytics: *telemetry*. Telemetry is data obtained over a distance, and is typically digital, but in principle any transmitted signal is telemetry. In the case of digital games, a common scenario sees an installed game client transmitting data about user-game interaction to a collection server, where the data is transformed and stored in an accessible format, supporting rapid analysis and reporting.
2. **Users:** Data on user behavior is arguably one of the most important sources of intelligence in game analytics, and user-oriented analytics is one of the key application areas of game analytics. Users in this context have a dual identity, as players of games and as customers. However, game analytics also covers areas such as production and technical performance, but these are less comprehensively covered in this book (but see for example Chaps. 6 and 7).

One of the main current application area of game analytics is to *inform Game User Research (GUR)*, which the chapters in this book also reflect. GUR is the application of various techniques and methodologies from e.g. experimental Psychology, Computational Intelligence, Machine Learning and Human-Computer Interaction to evaluate how people play games, and the quality of the interaction

between player and game. This is a big topic in game development in its own right (see e.g. Medlock et al. 2002; Pagulayan et al. 2003; Isbister and Schaffer 2008; Kim et al. 2008). The practice of GUR follows many of the same tenets as user-product testing in other ICT sectors, but with a general focus on the user experience which is paramount in game design (Pagulayan et al. 2003; Laramée 2005). Essentially, GUR is a form of game analytics because the latter covers all aspects of working with data in games contexts; but, game analytics is more than GUR. Where GUR is focused on data obtained from users, game analytics consider all forms of business intelligence data in game development and research.

This chapter is intended to lay the foundation for the book and provide a very basic introduction to game analytics. It is focused on describing the basic terminology of the domain with a specific emphasis on user behavior analytics. The chapter is structured in sections, as follows:

- **Section 2.2** lays out key terms and concepts in game analytics
- **Section 2.3** discusses the fundamental considerations guiding the selection of which user behaviors to track, log and analyze
- **Section 2.4** outlines the basics for collection and application of game telemetry data and the knowledge discovery process in game analytics.

Throughout the chapter, references are provided to other chapters in the book where topics introduced here are treated in more depth.

On a final note, this chapter does not go into direct detail on the *benefits* of applying game analytics to game development and research. This topic is the focus of Chap. 3, which details the benefits to all the main groups of stakeholders involved, e.g. designer and user research. Game analytics: key terminology.

There are many different kinds of data that can form the input streams in game analytics, and thus game BI. However, as mentioned above, this book is generally, but not exclusively (e.g. Chaps. 21 and 22), focused on **telemetry**.

2.1.1 Telemetry

The collection and application of telemetry has a history dating back to the nineteenth century where the first data-transmission circuits were developed, but today the term covers any technology that permits measurement over a distance (derived from Greek: tele=remote; metron=measure). Common examples include radio wave transmission from a remote sensor or transmission and reception of information via an IP network. **Game telemetry** is the term we use to denote any source of data obtained over distance, which pertain to game development or game research. There are many popular applications of telemetry in games, including remote monitoring and analysis of game servers, mobile devices, user behavior and production. The source of telemetry most strongly represented in this book is user telemetry, i.e. data on the behavior of users (players), for example on their interaction with games, purchasing behavior, physical movement, or their interaction with other users or

applications (Thompson 2007; Drachen and Canossa 2011; Mellon 2009; Bohannon 2010; Fields and Cotton 2011).

Game telemetry data can be thought of as the raw units of data that are derived remotely from somewhere, for example an installed client submitting data about how a user interacts with a game, transaction data from an online payment system or bug fix rates. In the case of user behavior data, code embedded in the game client transmits data to a collection server; or the data is collected from game servers (as used in e.g. online multi-player games like *Fragile Alliance* (Square Enix, 2007), *Quake* (id Software, 1996+) and *Battlefield* (EA, 2002)) (Derosa 2007; Kim et al. 2008; Canossa and Drachen 2009).

The actual data being transmitted follow different naming conventions depending on the field of research or application domain that people are applying the data to. This can cause some confusion when reading research articles on game analytics. The essence is that telemetry is measures of the *attribute of objects* (or *items*). Objects in this case should be understood broadly – an object can be virtual objects, people, processes, etc. – anything that has one or more measureable attributes. For example, the location of a player character as it navigates a 3D environment. In this case the location is the attribute, the player character the object. Conversely, the length of customer service calls generated from a newly released patch in an MMORPG sees the length of the calls as the attribute of the customer service calls.

In order to work with telemetry data, the attribute data needs to be *operationalized*, which means having to decide a way to express the attribute data. For example, deciding that the locational data tracked from player characters (or mobile phone users) should be organized as a number describing the sum of movement in meters. Operationalizing attribute data in this way turns them into variables or features – the term varies depending on the scientific field. In Experimental Psychology the term *variable* is usually used, and thus this is the term that is generally seen in articles and conference presentations on telemetry used in game user research. In Computer Science the term *feature* is often used, and thus this is the term used in data mining articles. This is just a general guideline – naming conventions vary considerably because game analytics is not a domain with established standards, so care must be taken when consulting the literature on game analytics (such as it is). Finally, variables/features have a specific *domain*. The domain is the set of all possible values – defining the domain is essentially what operationalizing attribute data is all about. For example, a binary domain allows only two values (e.g. 0 or 1).

2.1.2 Game Metrics

Raw telemetry data can be stored in various database formats (see Chaps. 6, 7 or 12), which are ordered in such a way that makes it possible to transform the data into various interpretable measures, such as average completion time as a function of individual game levels, average weekly bug fix rate, revenue per day, number of

daily active users, and so forth (see Chaps. 4 and 12). These are called **game metrics**. Game metrics are, in essence, interpretable measures of something. They present the same potential advantages as other sources of BI, i.e. support for decision-making in companies. Metrics can be variables/features and vice versa, or more complex aggregates or calculated values, for example the sum of multiple variables/features. To take an example: telemetry data from a shooter like *Quake* could include data on the location of the player avatar in the virtual environment, the weapons used, and information on whether every shot hits or misses, etc. These are different attributes, and they can be converted into variables/features such as “number of hits” or “number of misses” with a domain from 0 to 1,000 (with 1,000 being the biggest number of hits scored for a specific level). In turn, these simple variables/features can form the basis for analysis, e.g. calculating the hit/miss ratio for each level or map in *Quake* (e.g. “hit/miss ratio is 1.2 on average for the “Albatross” map”). An alternative is to use the variables/features “playerID”, “session length” and “points scored” to calculate the metric “points scored per minute” for each player. These kinds of measures, which are based on calculations involving several variables/features, are usually referred to as “game metrics”. However, there is no standard terminology widely accepted in game analytics, so be prepared for variations.

Additionally, it is important to note that most types of analysis and analytics software do not separate between a simple variable/feature or metric, or a more complex metric – when it comes to inputting measures into an analysis, they will follow the same naming standard as specified by the software. For example, in the statistics package SPSS (or PASW in newer generations) all measures of an object or objects are called “variables”. It does not matter whether this variable is a simple operationalization or a number calculated using a dozen such variables.

Metrics are usually calculated as a function of something. The typical unit is time, but can also be game build (version), country, progression in a game, or number of players or players’ ID, to name a few. All metrics are bound to some sort of timeframe, and this will always be from a past period – we cannot (yet) collect telemetry from the future. Telemetry based on past performance is generally referred to a “rear-view data”, and form the basis of traditional BI. However, it is possible to run predictive analyses based on historical data, which can generate metrics for future behavior, e.g. expected sales figures, expected churn rate, expected number of players, expected behavior of specific user groups, etc. However, these will always be based on predictions with a specific uncertainty attached, whereas collected telemetry data – if collected correctly – are facts.

To sum up, and provide a tentative and sufficiently broad definition, a *game metric* is a quantitative measure of one or more attributes of one or more objects that operate in the context of games. Translated into plain language, this definition clarifies that a game metric is a quantitative measure of something related to games. For example, a measure of how many daily active users a social online game has; a measure of how many units a game has sold last week; a measure of the number of employee complaints the past year; task completion rates in a production team for a specific title, etc. – are all game metrics, because they relate directly to some aspect of one or more games.

Conversely, metrics that are unrelated to the games context, for example the revenue of a game development company last year, the number of employee complaints last month, etc., are *business metrics*. The distinction can be blurry in practice, but is essential to separate what is purely business metrics with those metrics that relate to games, of which a number are unique to game development (in how many other IT sectors can “number of orcs killed per player” be a business metric?).

While the term game metrics has become something of a buzzword in game development in recent years, metrics have arguably been around for as long as digital games have been made, but the application of game telemetry and game metrics to drive data-driven design and development has expanded and matured rapidly in the past few years across the industry.

2.1.3 *Non-Telemetry-Based Metrics*

The term game metrics is often used as a synonym for measures based on operationalized game telemetry data, but it is worth noting that a game metric does not need to be derived from telemetry data. The connection between telemetry and game metrics is commonly made in game development due to the inspiration of the use of the term “metric” in web analytics and mobile analytics, which have been among the primary inspirational sources for game analytics.

A game metric is a quantitative measure of something related to games, but this does not specify that a particular method (i.e. telemetry) has to be used to obtain the measure. For example, the “average completion time” for a specific game level during a ten-person user test can be measured using a stopwatch or obtained via telemetry software. This does not change the fact that both resulting measures are metrics (but using a stopwatch introduces a potential problem with measurement accuracy). In this book, the term game metric is generally used for telemetry-derived measures, but as detailed in e.g. Chaps. 21 and 22, metrics can be derived from other sources of data.

2.1.4 *Game Metrics: Types and Classes*

Mellon (2009) categorized game metrics into three types, based on an expansion and slight redefinition of which the following categories of game metrics can be defined:

1. **User metrics:** (labeled “player metrics” in Mellon 2009) These are metrics related to the people, or users, who play games, from the dual perspective of them being either *customers*, i.e. sources of revenue or *players*, who behave in a particular way when interacting with games. The first perspective is used when calculating metrics related to revenue, e.g. average revenue per user (ARPU), daily active users (DAU) or when performing analyses related to revenue, e.g. churn analysis, customer support performance analysis or micro-transaction

analysis (see Chaps. 4 and 12). The second perspective is used for investigating how people interact with the actual game system and the components of it and with other players, i.e. focusing on in-game behavior. Examples of metrics are: total playtime per player, average number of in-game friends per player or average damage dealt per player; and common analyses include time-spent analysis, trajectory analysis, or social networks analysis (Chaps. 17, 18 and 19). The data used to generate player metrics typically originate in telemetry, notably from game clients, game servers or online payment processing tools (Chaps. 6 and 7).

The vast majority of the published knowledge about game analytics is based on player metrics, and this book is also biased towards the application of player metrics for game development. This focus on player metrics is driven at least in part by the increased focus on Game User Research (GUR) (see below and Chaps. 16, 21, 22, 25, 26 and 27 or 31 and 32 for a specific view on metrics and learning games) and the increasing popularity of social online games (Chap. 4).

2. **Performance metrics:** These are metrics related to the performance of the technical and software-based infrastructure behind a game, notably relevant for online or persistent games. Common performance metrics include the frame rate at which a game executes on a client hardware platform, or in the case of a game server, its stability. Performance metrics are also used when monitoring changing features or the impact of patches and updates on how well the client executes. A simple performance metrics known since the first game was programmed is the number of bugs found – per hour, day, week or any other timeframe. Performance metrics are heavily used in QA to monitor the health of a game build. It is also one of the most mature areas of game analytics, because the methods employed are derived from traditional software performance and QA techniques and strategies. See Chaps. 6, 7 and 23 for more on performance metrics.
3. **Process metrics:** These are metrics related to the actual process of developing games. Game development is to a smaller or greater degree a creative process, which – similar to other creative areas in IT – has necessitated the use of agile development methods. In turn, this has prompted the development of ways of monitoring and measuring the development process. For example, by combining task size estimation with burn down charts, or measuring the average turnaround time of new content being delivered, type and effect of blocks to the development pipeline, and so forth. Similar to performance metrics, a number of process metrics and the associated management and monitoring methods are adopted and/or adapted from the methods and strategies in use outside the games sector. See Chaps. 6, 7 and 23 for more on process metrics.

2.1.5 A Closer Look at User Metrics

“You are no longer an individual, you are a data cluster bound to a vast global network” – trailer for the game “Watch Dogs”(Ubisoft) presented at E3 in 2012

The above quote is pretty spot on when it comes to how game analytics view users in games – they are clusters of data about the attributes of a particular object (the player), and its connection to the larger “network” of the game. User metrics is a common source of business intelligence in a range of sectors, and this is also the case for game development and research. The vast majority of knowledge published in the past 5 years on game analytics is based on user metrics, and especially user behavior telemetry. This is not surprising given that the users (players) are alpha and omega for the success of a games title – games are products that are focused on delivering user experience, and being able to analyze how users interact with games is a prime source of information about the degree of success of a games’ design to deliver engaging experiences (Medlock et al. 2002; Kim et al. 2008; Nacke and Drachen 2011). User metrics therefore deserve a closer inspection.

A key feature of games – whether digital or not – is that they are **state machines**. What this means is that during play, a person creates a continual loop of actions and responses which keep the game state changing (Salen and Zimmerman 2003). The game engages the user and often loops the player through the same steps over and over again, keeping the user engaged over a period of time. This period of time arguably varies, but compared to e.g. purchasing a product from an online store, a game session takes longer time and generates a lot more actions from the user and reactions from the system – i.e. more state changes. This means that they generate more user-behavior data than most software applications, with terabytes of data easily being accumulated in a brief period of time (Drachen and Canossa 2011; Weber et al. 2011). This goes for both perspectives of the user: customer and player.

User metrics derived from games have been classified by their applicability across games by considering three levels of applicability: **generic metrics**, which apply across all digital games (total playtime per player, number of started game sessions); **genre specific metrics**, which are applicable to a specific genre, e.g. Role-Playing Games (RPGs) (character progression, number of quests/missions completed), and **game specific metrics**, which are specific to individual games, i.e. unique features e.g. the average number of white tarantulas killed in *Tomb Raider: Underworld* (Eidos Interactive, 2008), average number of times players chose each of the three endings in *Mass Effect 3* (Electronic Arts, 2012). This system of classification is useful for research purposes, but a more development-oriented classification system, which serve to funnel user metrics in the direction of three different classes of stakeholders, is suggested here (shown in Fig. 2.1).

- **Customer metrics:** Covers all aspects of the user as a customer, e.g. cost of customer acquisition and retention. These types of metrics are notably interesting to professionals working with marketing and management of games and game development.
- **Community metrics:** Covers the movements of the user community at all levels of resolution, e.g. forum activity. These types of metrics are useful to e.g. community managers.
- **Gameplay metrics:** Any variable related to the actual behavior of the user as a player – inside the game, e.g. object interaction, object trade, and navigation in

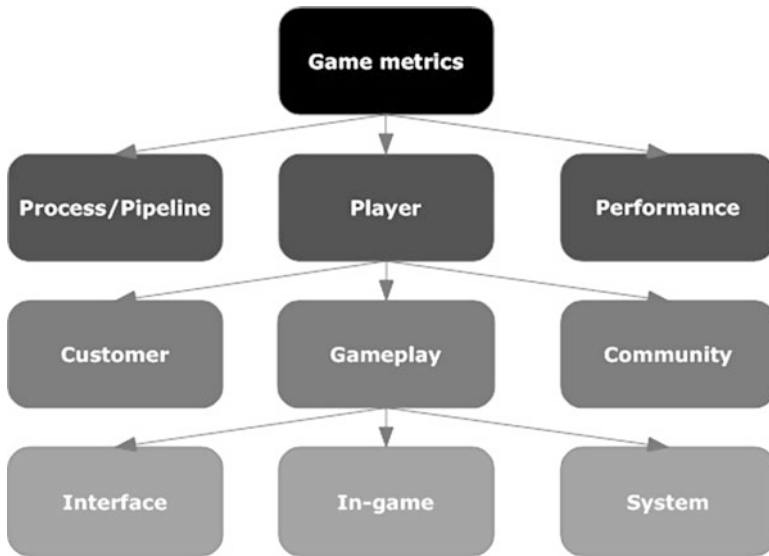


Fig. 2.1 Hierarchical diagram of game metrics emphasizing user metrics

the environment. Gameplay metrics are the most important to evaluate game design and user experience, but are furthest from the traditional perspective of the revenue chain in game development, and hence are generally under prioritized. These metrics are useful to professionals working with design, user research, quality assurance, or any other position where the actual behavior of the users is of interest.

2.1.5.1 Customer Metrics

As a **customer**, users can download and install a game, purchase any number of virtual items from in-game or out-of-game stores and shops, spending real or virtual currency, over shorter or longer timespans. At the same time, customers interact with customer service, submit bug reports, requests for help, complain, or otherwise interact with the company. Users can also interact with forums, whether official or not, or any other kind of social interaction platform, from which information about the users, their play behavior and how satisfied they are with the game, can be mined and analyzed (see Chap. 7). Customers also have properties. They live in specific countries, generally have IP-addresses, and sometimes we details about them such as their age, gender and email address. Combining this kind of demographic information with behavioral data can provide powerful insights into a games' customer base. Chapter 4 describes a number of examples of customer metrics.

2.1.5.2 Community Metrics

Players interact with each other. This interaction can be related to gameplay – e.g. combat or collaboration through game mechanics – or social – e.g. in-game chat. Player-player interaction can occur in-game or out-of-game, or some combination thereof. For example, sending messages bragging about a new piece of equipment using a post-to-Facebook function. In-game, interaction can occur via chat functions, out-of-game via live conversation (e.g. using Skype) or via game forums.

These kinds of interactions between players form an important source of information, applicable in an array of contexts. To take an example, social networks analysis of the user community in a free-to-play (F2P) game can reveal players with strong social networks, i.e. players who are likely to retain a big number of other players in the game via creating a good social environment. A good example is guild leaders in MMORPGs. Mining chat logs and forum posts can provide information about problems in a game’s design. For example, data mining datasets derived from chat logs in an online game can reveal bugs or other problems (see Chap. 7 for an example). Monitoring and analyzing player-player interaction is important in all situations where there are multiple players, but especially in games that attempt to create and support a persistent player community, and which have adopted an online business model, e.g. many social online games and F2P games. These examples are just the tip of a very deep iceberg, and the collection, analysis and reporting on game metrics derived from player-player interaction is a topic that could easily take up a book on its own. See Chaps. 4, 7 and 21 for more on this topic.

2.1.5.3 Gameplay Metrics

This sub-category of the user metrics is perhaps the most widely logged and utilized type of game telemetry currently in use in the industry. Gameplay metrics are measures of player behavior, e.g. navigation, item- and ability use, jumping, trading, running and whatever else players actually do inside the virtual environment of a game (whether 2D or 3D). Five types of information can be logged whenever a player does something – or is exposed to something – in a game: *What is happening? Where is it happening? At what time is it happening?* In addition, when multiple objects (e.g. players) interact: *to whom is it happening?*

Gameplay metrics are particularly useful to game user research for informing game design. They provide the opportunity to address key questions, including whether any game world areas are over- or underused, if players utilize game features as intended, or whether there are any barriers hindering player progression. This kind of game metrics can be recorded during all phases of game development, as well as following launch (Isbister and Schaffer 2008; Kim et al. 2008; Lameman et al. 2010; Drachen and Canossa 2011).

As a player, users can generate thousands of behavioral measures over the course of a just a single game session – every time a player inputs something to the game system, it has to react and respond. Accurate measures of player activity can include dozens of

actions being measured per second. Consider, for example, player in a typical fantasy MMORPG like *World of Warcraft* (Blizzard, 2003): measuring user behavior could involve logging the position of the player's character, its current health, mana, stamina, the time of any buffs affecting it, the active action (e.g. running, swinging an axe), the mode (in combat, trading, traveling, etc.), the attitude of any MOBs towards the player, the player character name, race, level, equipment, currency etc. – all these bits of information flowing from the installed game client to the collection servers.

From a practical perspective (e.g. for naming different groups of metrics in a way that makes them easily searchable), it can be useful to further subdivide gameplay metrics into the following three categories:

- **In-game:** Covers all in-game actions and behaviors of players, including navigation, economic behavior as well as interaction with game assets such as objects and entities. This category will in most cases form the bulk of collected user telemetry.
- **Interface:** Includes all interactions the user (player) performs with the game interface and menus. This includes setting game variables, such as mouse sensitivity, monitor brightness.
- **System:** System metrics cover the actions game engines and their sub-systems (AI system, automated events, MOB/NPC actions, etc.) initiate to respond to player actions. For example, a MOB attacking a player character if it moves within aggro range, or progressing the player to the next level upon satisfaction of a pre-defined set of conditions.

To sum up, the sheer array of potential measures from the users of a game (or game service) is staggering, and generally analysts working in game development try to locate the most essential pieces of information to log and analyze. This selection process imposes a bias but is often necessary to avoid data overload and to ensure a functional workflow in analytics (for more on this topic see Chaps. 3, 4, 6, 7, 9, 12 and 14).

2.1.6 Example Gameplay Metrics Across Game Types

Up to this point the discussion about user attributes has been at a fairly abstract level, because it is nigh-on impossible to develop classes of which user metrics it makes sense to develop in which types of games. This not just because games do not fall within neat design classes (games share a vast design space but do not cluster at specific areas of it), but also because the rate of innovation in design is high, which would rapidly render recommendations invalid. In this section some examples of useful gameplay metrics are provided for different game genres. Despite being nebulous, genre definitions are commonly used to provide e.g. readers of game reviews some idea about which type of game we are dealing with. For example, labeling *Skyrim* (Bethesda Softworks, 2011), *Deus Ex Human Revolution* (Eidos Interactive, 2000) and *Diablo III* (Blizzard Entertainment, 2012) as Role-Playing Games, due to

the ability of the user to modify the character played during the game, irrespective of the many other differences in their gameplay and style. Genres make for useful terms when defining what “school” of mechanics drive a game.

In essence, it is the mechanics (and thus by inference genre, but keeping in mind that genres are nebulous), and the underlying business model (e.g. traditional one-shot vs. F2P) which determines what types of player telemetry that can be logged and analyzed.

2.1.6.1 Action Games

Action games are generally focused on quick reflexes, accuracy, timing etc., over to more explorative-heavy games. Usually single character/avatar played. Examples: Pinball games, racing, FPS’ and TPS’.

Useful gameplay metrics: In general anything that relates to the reflex-based mechanics.

First-Person Shooters (FPS)

First-Person Shooters are shooter games, i.e. focused on combat involving projectile weapons of some kind, with the camera looking out of the eyes of the player. Fast paced games, reflex-based play, can include strategic elements, heavily reliant on engagement. Examples: *Unreal* (GT Interactive, 1998), *Quake* (GT Interactive, 1996), *Halo* (Microsoft Studios, 2001). Note how team-based FPS’ like *Team Fortress 2* (Valve, 2010) track a wealth of player behaviors and provide them back to the players.

Useful gameplay metrics: Weapon use, trajectory, item/asset use, character/kit choice, level/map choice, loss/win [quota], heatmaps, team scores, map lethality, map balance, vehicle use metrics, strategic point captures/losses, jumps, crouches, special moves, object activation. AI-enemy damage inflicted+trajectory. Possibly even projectile tracking.

Third-Person Shooters (TPS)

Third-Person Shooters are shooter games, i.e. focused on combat involving projectile weapons of some kind, with the camera from a third-person perspective relative to the player avatar. Includes shoot’em up-games, arcade-style games where the player controls a central avatar who kills massive numbers of enemies. Fast paced games, reflex-based play, can include strategic elements. Examples: *Project X* (Team17 Software, 1992), *Starfighter* (Micros, 1984), *Aerial Command* (Croft Soft Software, 1994).

Useful gameplay metrics: as for FPS + camera angle, character orientation.

Racing

Racing games are games where the player controls a vehicle. Usually reliant on reflex gameplay and some strategic thinking.

Useful gameplay metrics: Track choice, vehicle choice, vehicle performance, win/loss ratio per track and vehicle, completion times, completion ratio per track and player, upgrades [if possible], color scheme [if possible], hits, avg. speed different types of tracks/track shapes.

2.1.6.2 Adventure Games

Maybe two different genres – Adventure and Action-adventure – but exceptionally hard to separate. Incredibly varied – usually single-player, focused on exploration and puzzle-solving, but can also include combat, although normally not reliant on reflex-based play. Often heavy story element. Includes interactive stories. Puzzle heavy. Examples: *Deus Ex: Human Revolution* (Square Enix, 2011), *Tomb Raider: Underworld* (Square Enix, 2008). Pattern analysis is highly useful (see Chap. 12).

Useful gameplay metrics: story progression [e.g. node based], NPC interaction, trajectory, puzzle completion, character progression, character item use, world item use, AI-enemy performance, damage taken and received + source (player, mob).

2.1.6.3 Arcade

Simple mechanics, fast-paced play, generally game is never completed. Example: *Pac-Man* (Atari, 1981), *Asteroids* (Atari, 1981).

Useful gameplay metrics: trajectory, powerup usage, special ability usage, session length, stages completed, points reached, unlocks, opponent type damage dealt/received, player damage dealt/received [as applicable].

2.1.6.4 Beat'em Up

Fighting game, generally restricted to one player controlling one avatar in combat with another, but can be multi-player beyond two people. Generally players control a “humanoid” avatar. Examples: *Double Dragon* (Activision, 1988), *Tekken* (Namco Bandai, 1995).

Useful gameplay metrics: Character selection, ability use, combo use, damage dealt, damage received (per ability, character etc.), weapon usage, arena choice, win/loss ratio as a feature of character, player skill profiles.

2.1.6.5 Family Games

A game designed to be played by both adults and children together. Example: *Mario Kart* (Nintendo, 2011), *Buzz!* (Sony, 2005) Includes partygames.

Useful gameplay metrics: varies substantially – subgame selection, character/avatar selections, game mode used, in-game selections, asset use, number of players, etc. form some of the possibilities.

2.1.6.6 Fitness Games

Also called exergames. A game designed to improve people’s fitness. Often played in combination with various hardware accessories. Examples: *Yourself Fitness* (Respondesign, 2008), *Wii Fit* (Nintendo, 2007), *Dance Dance Revolution* (Konami, 2001).

Useful gameplay metrics: session length, calories burned, exercises chosen, match between exercises shown and player actions, player accuracy in performing exercises, total playtime over X days, player hardware/exercise equipment [usually registered], player demographics [usually entered during profile creation], music tracks selected, backgrounds selected, avatar selection, powerups/content unlocked [common feature], total duration of play per user.

2.1.6.7 Music Games

Also called audiogames. A game where the players sing or where the gameplay is otherwise heavily reliant on music-related mechanics. Commonly challenge the player to follow sequences of movement or develop specific rhythms. Examples: *Singstar* (Sony, 2004).

Useful gameplay metrics: Points scored, song/track chosen, match with rhythm/auditory mechanics, difficulty setting, track vs. difficulty, track vs. errors, track vs. choices.

2.1.6.8 Platformer Games

A game focused on navigation in 2D or 3D space along platforms. Examples: *Mario* (Nintendo, 1983-), *Sonic* (Sega, 1992-), *Giana Sisters* (a.k.a. The Great Giana Sisters, Rainbow Arts, 1987).

Useful gameplay metrics: jumping, progression speed, items collected, powerups/abilities used, AI-enemy performance, damage taken + sources of damage.

2.1.6.9 RPGs

Role Playing Games are extremely varied – can be any other genre but includes crucially the ability for the player to develop the avatar/character/-s being controlled. Examples: *Diablo* (Blizzard Entertainment, 1996), *Dragon Age: Origins* (EA, 2009), *Mass Effect* (BioWare, 2007), *Eye of the Beholder* (Strategic Simulations, 1991-). Temporal and spatial analysis can be useful, see Chap. 19 for more on analysis of RPGs.

Useful gameplay metrics: character progression, quest completions, quest time to complete, asset use (resources), character ability/item use [including context of use], combat statistics, AI-enemy performance, story progression [including choices], NPC interactions [e.g. communication], ability/item performance, damage taken+sources of damage, cutscene viewed/skipped, items collected [including spatial info].

2.1.6.10 Simulation

Simulation: A very diverse category of games, where the main focus is on simulating some aspect of life or fiction, from constructing and managing cities in *SimCity* (Maxis, 1989-) to simulating life in *Spore* (EA, 2008) and *Evo* (Enix Corporation, 1992), or vehicles from cars to air planes, including combat simulators.

Useful gameplay metrics: Very hard to predict due to the sheer variety in simulation games. Asset use would be important, but depends on the specifics of the game.

2.1.6.11 Sports Games

Any game where the main focus is on the execution of sports activities. Examples: *FIFA World Manager* (Ubisoft Entertainment, 1998), *Madden NFL* (EA, 1993-), *Wii sports* (Nintendo, 2006).

Useful gameplay metrics: match types, win/loss ratios, team selection, color schemes, country chosen, management decisions [if game includes management aspects], in-match events [e.g. goal scored, fouls, tackles, length of hit], item use [e.g. club type], heatmap [density of player time spent on sections of the field], team setup/strategy, player [in-game] selection, player commands to team/team members.

2.1.6.12 Strategy Games

Can be broadly divided into either real-time or turn-based strategy games (e.g. *Starcraft* vs. *Civilization V*) (Blizzard, 1998; 2K Games, 2010). The gameplay is focused on strategic planning and plan execution, and often the player controls multiple avatars, e.g. units. More specialized strategy games include smaller groups of units to control and a TPS/FPS view. Includes the small category of “god games”

and “puzzle games” as well as tower defense games. Metrics choices vary generally depending on whether the game is a real-time strategy game (RTS) or turn-based (TBS). Spatial analysis can be useful for these types of games see Chap. 19 for an example.

Useful gameplay metrics: all features related to player strategy and control. Generally two types of things players can build: building and units. Selections and order of selection are crucial metrics. Commands given to units, upgrades purchased, trajectory, win/loss ratio, team/race/color selection, maps used, map settings, match/game settings (usually strategy games have some settings that affect the core mechanics). Race/aspect/team chosen, time spent on building tasks vs. unit tasks.

2.1.7 Tracking Strategies

The transmission of a piece of information via a telemetry system – irrespective of whether this is in the context of user, process or performance measures – in games can occur in three fundamental ways:

1. **Event:** A pre-specified event occurs, for example, a user starts a game, a designer submits a bug fix request, a unit of a game is sold, a player fires a weapon, buys an item, etc. – any action initiated by a person or system forms an event. Event-based telemetry is based on tracking such actions and transmitting this information to a collection server.
2. **Frequency:** Rather than being triggered by the occurrence of a specific event, information can be recorded following a specific frequency. For example, when tracking the trajectory of player avatars through virtual environments, we can record the location of the avatar once per second, as a compromise between precision and bandwidth constraints. Frequency-based recording of telemetry is generally used when the attribute of the object being tracked is always present, e.g., a player character in an MMORPG always has a position in the world when playing.
3. **Initiated:** Sometimes the game analyst wants to enable and disable the tracking of a specific attribute, rather than having a telemetry system autonomously submitting tracked information based on some pre-defined command. For example, it may not be necessary to record player avatar trajectories all the time, but only when updates or patches are pushed to the users. Having the ability to turn on and off recording of specific attributes can be useful in these situations.

There are different strategies available for the recording, transmission and storage of game telemetry. For example, sampling can be employed to reduce the overall amount of data being collected and thereby to reduce costs of running analyses. This topic is described in Chap. 9. Similarly, there are different options for how to physically handle the client-side recording and transmission of telemetry to collection servers, a topic discussed in Chaps. 6, 7 and 12.

2.2 Ethics and Privacy

A key issue when working with user telemetry is the question of individual privacy and ethics. Current technology allows for the collection of detailed information on users from digital games, and combined with information harvested via collaboration with third parties, highly detailed patterns of behavior can be mined, e.g. information about the habits and preferences of individuals. Collected behavioral telemetry can be used to generate player profiles, and correlated with personal data forms a means of targeting players with marketing messages that are highly specific. The existence of such datasets is controversial given their confidential nature and the potential illegal access and use. Because data are valuable, they are also traded, and this can lead to user information migrating to the hands of people who will employ the knowledge unethically.

Currently the typical practice in the game industry is to keep the user (or consumer) data confidential and not sell or share this data. Furthermore, most analysis is run on anonymized data, so the identities of the users are not shown to the analyst working with the data, although basic information such as demographics might be known. It is however not the norm that users are clearly informed that their behavior is being tracked, and there is rarely a chance to opt out of tracking and still play the game in question. The digital analytics association has developed the Web Analyst Code of Ethics, which are directly applicable to game analytics (<http://www.digitalanalyticsassociation.org/?page=codeofethics>), however, there is currently no widely agreed – upon standard in game analytics, and ethics therefore remain a largely grey and undefined area in the field. Some of these issues have been discussed in the interview with the independent developer Niffas, Chap. 20.

2.3 Selecting User Behaviors

Having covered the basic categories of game metrics the next question that arises is: given the array of possible variables/features to track from a digital game, which of these should we track? There is no one answer to this question. Like all other applications of BI, game analytics is a highly context-dependent process, perhaps especially so for computer games, because of the substantial variation in design, business models, target audience, revenue drivers, value chains etc. However, as noted above, games that share design features, e.g. free-to-play social online games for *Facebook*, will likely share metrics related to these shared features that are useful across these games – but not outside of them.

In comparison, *process* and *performance* metrics are more generalizable across games and companies, because there is a substantial overlap in the methods employed in game development across the industry, e.g. a common use of agile frameworks, similar marketing strategies, and so forth (see any book on game development and –management for more on game production) (e.g. Laramee 2005). When selecting user metrics, especially for titles with complex gameplay and thus

hundreds of possible user actions and -interactions, the question becomes more difficult to answer. In this section we outline some of the fundamental considerations in *feature selection* (selecting what user behaviors to track and analyze), more in-depth discussion can be found in Chap. 13. For feature selection and key metrics in social online games/F2P specifically, see Chaps. 4 and 35 for an interview with Junebud or Fields and Cotton (2011).

2.3.1 *Balancing Cost and Benefit*

User-oriented game analytics can have a variety of purposes, but can broadly be divided into:

- *Strategic analytics*, which target the global view on how a game should evolve based on analysis of user behavior and the business model.
- *Tactical analytics*, which aim to inform game design at the short-term, for example an A/B test of a new game feature.
- *Operational analytics*, which target analysis and evaluation of the immediate, current situation in the game. For example, informing what changes should be made to a persistent game to match user behavior in real-time.

Operational and tactical analytics to an extent deal with technical and infrastructure issues, whereas strategic analytics is more focused on merging user telemetry data with other user data and/or market research.

The first thing to be aware of when deciding on a strategy for how to approach user telemetry is the existence of these three types of user-oriented game analytics, and the kinds of input data they require. The second aspect to consider is the diminishing returns on the logging of behavioral telemetry.

In a situation with infinite resources, it is possible to track, store and analyze every single user-initiated action – every fraction of a move of an avatar, every button press, all purchases made, every single chat message, all the server-side system information – even all keystrokes. Doing so will likely cause bandwidth issues, and will require substantial resources to add the message hooks into the game code, but in theory, this brute-force approach to game analytics is possible. However, it leads to very large datasets, which in turn leads to huge resource requirements in order to transform and analyze them (Han et al. 2011; Kim et al. 2008; Drachen et al. 2009). For example, tracking the weapon type, range, damage done, target, whether the target was killed or not, the weapon modifications chosen by the player, the position of the player and target, the trajectory of the bullet, etc. will provide the possibility for a very in-depth analysis of weapon use in an FPS. However, the key metrics to calculate in order to evaluate weapon balancing could just be range, damage done and the frequency of use of each weapon. Adding a number of additional variables/features may not add any new relevant insights, or may even add noise or confusion to the analysis. Similarly, it may not be necessary to log behavioral telemetry from all players of a game, but only a percentage. This is of course not the case when it comes to sales records (see Chap. 9 for more on sampling).

In general, if selected correctly, the first variables/features that are tracked, collected and analyzed will provide a lot of insights into user behavior. As more and more detailed aspects of user behavior are tracked, costs of storage, processing and analysis increase but the rate of added value from the information contained in the telemetry data diminishes.

What this means is that there is a cost-benefit relationship in game telemetry, which basically describes a simplified **theory of diminishing returns** (from Economic theory): Increasing the amount of one source of data in an analysis process will yield a lower per-unit return.

A classic example in Economic literature is adding fertilizer to a field. In an unbalanced system (under-fertilized), adding fertilizer will increase the crop size, but after a certain point this increase diminishes, stops and may even reduce the crop size. Adding fertilizer to an already balanced system does not increase crop size, or may reduce it.

Fundamentally, game analytics follows a similar principle. An analysis can be optimized up to a specific point given a particular set of input features/variables, before additional (new) features are necessary. Additionally, increasing the amount of data into an analysis process may reduce the return, or in extreme cases lead to a situation of negative return due to noise and confusion added by the additional data.

There can of course be exceptions – for example, the cause of a problematic behavioral pattern, which decreases retention in a social online game, can rest in a single small design flaw, which can be hard to identify if the specific behavioral variables related to the flaw are not tracked.

2.3.2 User Experience Versus Monetization

The fundamental goal of game design is to create games that provide a good user experience. However, the fundamental goal of running a game development company is to make money. Aligning these two goals is vital, especially in the F2P game situation where there is no up-front investment from the customers.

In this situation, the underlying drivers for game analytics are twofold: (1) ensuring the user experience, in order to acquire and retain customers; (2) ensuring that the monetization cycle generates revenue.

It is possible to design F2P games that provide a good user experience but which absolutely fail in prompting the users to make any purchases. Similarly, it is possible to design a F2P which includes all the tricks in the book to make users to invest real-world money in the game. These two extremes have a hard time standing on their own, and therefore user-oriented game analytics must inform both design and monetization at the same time.

This approach is exemplified by companies who have been successful in the F2P marketplace, e.g. Zynga, Kiloo and Wooga, who use analysis methods like A/B

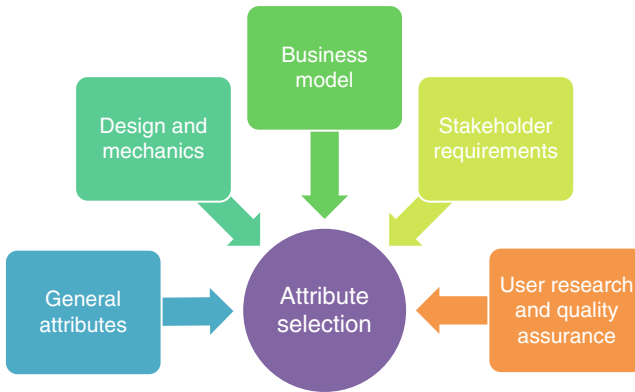


Fig. 2.2 The drivers of attribute selection for user behavior attributes. Given the broad scope of application of game analytics, a number of sources of requirements are in play

testing to evaluate whether a specific design change increases both retention and monetization (see Chaps. 4 and 5 for an interview with Zynga, Chap. 24 for an interview with Kiloo and Fields and Cotton 2011).

2.3.3 Feature Selection of User Behavior Telemetry

In real life we rarely have the resources to track and analyze all possible user behaviors, which necessitates an approach to analytics which considers cost-benefit relationships between on one side the resources required for tracking, storing and analyzing user telemetry/metrics, on the other hand the value of the insights obtained (Mellon 2009).

Following this line of reasoning, the minimum set of attributes that should be tracked, stored and analyzed about users in a computer game context is comprised of (Fig. 2.2):

1. **General attributes:** The attributes that are shared for users (as customers and players) across all games. These form the core metrics which can always be collected, for any computer game, e.g. when a user starts playing a game, stops playing, a userID, etc. For examples, see e.g. Chaps. 14, 18 and 19.
2. **Core mechanics/design attributes:** The essential attributes related to the core of the gameplay and mechanics of the game. For example attributes related to time spent playing, virtual currency spent, number of opponents killed, etc. Defining the core mechanics attributes should be based directly on the key gameplay mechanics of the game, and provide information that allows inferences to be made about the user experience. For example, whether players are progressing as planned, if flow is sustained, death ratios, level completions, point scores, etc. For examples, see e.g. Chaps. 14 and 17.

3. **Core business attributes:** The essential attributes related to the core of the business model (e.g. F2P) of the company. For example, logging every time a user purchases a virtual item, establishes a friend connection in-game, country of origin, recommends the game to a *Facebook* friend, attributes related to retention, virality and churn, etc. See Chaps. 4 and 12 for more on business-related metrics (or Fields and Cotton 2011).

In addition to these three, there can be an assortment of *stakeholder requirements* that need to be considered. For example, management or marketing may place a high value on knowing the number of Daily Active Users (DAU) (Chap. 4). Such requirements may or may not align with the categories mentioned above.

Finally, if there is any interest in using telemetry data for *user research/user testing* and *quality assurance* (e.g. recording crashes and crash causes, hardware configuration of client systems, and notably game settings), it may be necessary to augment to attributes on the list of features accordingly. Chapters 6 and 7 provides insights on this area.

When building the initial attribute set and planning the metrics that can be derived from them, it is vital that the selection process is as well-informed as possible, and includes all the involved stakeholders. This minimizes the need to go back to the code and embedding additional hooks at a later time – which is a waste that can be eliminated with careful planning. That being said, as the game evolves during production as well as following launch (whether a persistent game or through DLCs/patches), it will typically be necessary to some degree to embed new hooks in the code in order to track new attributes and thus sustain an evolving analytics practice.

Sampling is another key consideration. It may not be necessary to track every time someone fires a gun, but only 1% of these. Chapter 9 discusses sampling in detail and we will therefore not delve further on this subject here, apart from noting that sampling can be an efficient way to cut resource requirements for game analytics.

2.3.4 Pre-selecting Attributes

A final consideration is the extent to which attribute set selection can be driven by pre-planning, i.e. by defining the game metrics and analysis results (and thereby the actionable insights) we wish to obtain from user telemetry and select attributes accordingly. This is certainly possible to an extent, but the lack of an explorative component adds the risk of missing important patterns in user behavior that cannot be detected using the pre-selected attributes. This problem is exasperated in situations where the game metrics and analyses are also pre-defined – for example relying on a set of Key Performance Indicators (KPIs) – which eliminates the chance of finding any patterns in the behavioral data not detectable via the pre-defined metrics and analyses. In general, striking a balance between the two situations – tracking and analyzing everything vs. pre-selecting KPIs – is the best solution. See e.g. Chaps. 6, 7 and 14 for examples of pre-selection in practice. In Chap. 8, interview with Unity Technologies, the holy grail of dynamic feature tracking is discussed.

It is worth noting that when it comes to user behavior analytics, we are working with human behavior, which is notoriously unpredictable, at least in games contexts. This means that predicting user analytics requirements can be problematic, and forms the basis for the use of both explorative (i.e. we look at the user data to see what patterns they contain) and hypothesis-driven methods (i.e. we know what we want to measure and know the possible results, just not which one is correct), in e.g. Game User Research. These approaches are described in more detail in Chaps. 13 and 14, and by e.g. Pagulayan et al. (2003); Isbister and Schaffer (2008) and Drachen and Canossa (2011).

2.4 Telemetry Analysis and Reporting

In the above sections the fundamental terminology of game analytics has been introduced, and an overview presented of the different types of data that can form the input to the game analytics process. We now turn to the process of collecting and utilizing game telemetry. These are topics that are described in the remainder of this book, and this section will therefore briefly introduce the general steps in the game analytics process, and provide references to chapters where the different topics are treated; and the references and suggested readings in those chapters provide a guideline for further reading on the various topics. An important topic not covered here is how to integrate analytics in the business and culture of an organization. See Chaps. 6 and 7 for discussion on this topic.

The game analytics process follows the standard process for *knowledge discovery* in data (Berry and Linoff 1999; Larose 2004; Witten et al. 2011), which is widely used in data-driven analytics to discover useful knowledge from data. Knowledge discovery can be described in a number of phases or steps, which are fundamentally cyclic in nature, i.e. the result of an analysis cycle can feed into the next cycle. This is one way of continually optimizing the discovery process. The knowledge discovery process is described in Chap. 12, here we present a brief overview, with phases adapted to the context of game development and the focus on user telemetry as the data source (Fig. 2.3). The systems used to enable knowledge discovery are in Business Intelligence generally referred to as Decision Support Systems (DSS) or Knowledge Discovery Systems (KDS) depending on the specific aim.

1. **Attribute definition:** The first step in the process is defining the objectives, and the requirements, the result of the discovery process must fulfill. During this phase the user attributes to track are selected, as well as the tracking strategy (event, frequency or initiated). Domains for each attribute are defined, and *goals* for each domain defined. For example, it may be a goal that the maximum play-time for a game is set to 20 h (i.e. the game should not take any longer to complete). During this phase, strategies for balancing pre-defined metrics and results are balanced against the requirement for being able to explore and drill-down/ across/through datasets (see Chap. 12 for an introduction to game data mining, Chap. 14 for an introduction to explorative telemetry work).



Fig. 2.3 The phases of the standard knowledge discovery process adapted to the context of game analytics

2. **Data acquisition:** Once the attribute set has been defined, it is implemented in the telemetry system the company uses. If no such system exists, one will have to be either purchased or a service agreement entered. There are fundamentally three ways to obtain a telemetry system: (1) develop in-house, (2) purchase a license, or (3) purchase access to a software-as-a-service solution. There are at the time of writing about two-dozen companies worldwide offering telemetry solutions for games. Several of these are solutions developed for e.g. business analytics or web analytics, but are also applicable to some, and in a few cases all, types of games. There are unfortunately no comprehensive guides or reviews of telemetry providers currently available, and a degree of research is therefore needed to locate the solution best suits the requirements. Chapters 6 and 7 describe two examples of telemetry systems built at Bioware and Sony Online Entertainment respectively. Chapter 10 provides an example of a flexible, open-source RestAPI for tracking user telemetry from games, aimed at small-medium sized games.
3. **Data pre-processing:** During this step, incoming telemetry data are transformed and loaded into a database structure (see Chap. 12 for a short overview of SQL

and NoSQL solutions), from where they are accessible for analysis. Additionally, data are cleaned and otherwise made ready for analysis. Chapter 12 describes data pre-processing in general, and Chap. 7 provides more specific examples.

4. **Metrics development:** following pre-processing, the attribute data are transformed into variables/features and metrics. This can be done automatically (e.g. KPIs) or manually.
5. **Analysis and evaluation:** During this step, cases and features are selected as required by the analysis in question. Sampling can also be applied to minimize resource requirements (see Chap. 9). The chosen analysis is run and a model generated of the results (see Chap. 12). Furthermore, results are evaluated and it is checked if the model reaches the required objectives.
6. **Visualization:** The results are visualized in a way that is functional given the stakeholder they aimed at, following principals of knowledge visualization (Tufte 1983, is the classic text on representation and visualization of quantitative data). Chapters 18 and 19 describe visualization of user behavior telemetry in detail. Additionally, Chap. 17 focuses on spatial metrics and visualization.
7. **Reporting:** The discovered knowledge is presented to the relevant stakeholders, e.g. a designer. The reporting/presentation should be done in such a way that the stakeholders can understand, interpret and act on the result.
8. **Knowledge deployment:** The knowledge is deployed in the organization. This will often see the initiation of a new discovery cycle.

There is a lot more to say and write about the fundamental process of knowledge discovery as it applies to game analytics, but the above covers the basic steps. Several chapters in this book go into more detail with the different steps, and steps 7 and 8 are topics that more or less all chapters touch upon, because achieving action following analysis is a fundamental goal of industrial analytics.

2.5 Conclusions and Next Steps

In this chapter the bare-bones fundamentals of game analytics have been outlined. Important key terminology has been described setting the background for game analytics as a source of business intelligence in game development and game research. The benefits of adopting analytics have been outlined, which can be summed up the following: support for decision making – at all levels and areas of an organization (or a research project). In the above we have also discussed users and user behavior in some detail, as this is a topic of core interest in game analytics.

Finally, we introduced the different challenges in feature/variable selection, and the knowledge discovery process describing the journey from raw, untreated data to actionable insights. The remaining chapters in this book will go into more detail with the topics briefly introduced here, and beyond.

On a final note, the reference list below provides an excellent starting point for further reading on game analytics.

About the Authors

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. Magy worked collaboratively with *Electronic Arts*, *Bardel Entertainment*, and *Pixel Ante*.

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and the Royal Danish Academy of Fine Arts, Schools of Architecture, Design and Conservation. His doctoral research was carried out in collaboration with IO Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches. His work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he maintains an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio.

References and Next Steps

- Berry, M., & Linoff, G. (1999). *Mastering data mining: The art and science of customer relationship management*. New York: Wiley.
- Bohannon, J. (2010). Game-miners grapple with massive data. *Science*, 330(6000), 30–31.

- Canossa, A., & Drachen, A. (2009). *Patterns of play: Play-personas in user-centered game development*. DIGRA. London: DIGRA Publishers.
- Davenport, T. H., & Harris, J. G. (2007). *Competing on analytics: The new science of winning*. Boston: Harvard Business School Press.
- Derosa, P. (2007). 'Tracking player feedback to improve game design'. Gamasutra. Available at: http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_.php
- Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behavior in virtual environments. *International Journal of Arts and Technology*, 4(3), 294–314.
- Drachen, A., Canossa, A., & Yannakakis, G. (2009). Player modeling using self-organization in tomb raider: Underworld. In *Proceedings of IEEE Computational Intelligence in Games (CIG)* (pp. 1–8). Milan: IEEE Publishers.
- Fields, T., & Cotton, B. (2011). *Social game design: Monetization methods and mechanics*. Burlington: Morgan Kaufman Publishers.
- Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques, 3rd*. San Francisco: Morgan Kaufmann Publishers.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. Burlington: Morgan Kaufman Publishers.
- Jansen, B. J. (2009). *Understanding user-web interactions via web analytics*. San Rafael: Morgan & Claypool Publishers.
- Kim, J. H., Gunn, D. V., Schuh E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). Tracking Real-Time User Experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of the Computer-Human Interaction (CHI)* (pp. 443–451), Florence, Italy.
- Lameman, B. A., Seif El-Nasr, M., Drachen, A., Foster, W., Moura, D., & Aghabeigi, B. (2010) User studies – A strategy towards a successful industry-academic relationship. In *Proceedings of future play 2010* (pp. 1–9). Vancouver: ACM Publishers. doi:10.1145/1920778.1920798
- Laramee, F. E. (2005). *Secrets of the game business*. Hingham: Charles River Media.
- Larose, D. T. (2004). *Discovering knowledge in data: An introduction to data mining*. Hoboken: Wiley-Interscience.
- Law, E., Vermeeren, A. P. O. S., Hassenzahl, M., & Blythe, M. (2007). Towards a UX manifesto. In *Proceedings of the 21st British HCI group annual conference on HCI 2008: People and computers XXI: HCI ... but not as we know it*, (pp. 205–206), Florence, Italy.
- Luhn, H. P. (1958). A business intelligence system. *IBM Journal*, 2(4), 314.
- Medlock, M. C., Wixon, D., Terrano, M., Romero, R. L., & Fulton, B. (2002). Using the RITE method to improve products: A definition and a case study. In *Proceedings of the Usability Professionals Association*, Orlando, Florida.
- Mellon, L. (2009). Applying metrics driven development to MMO costs and risks. Versant Corporation, Tech. Rep.
- Nacke, L., & Drachen, A. (2011). Towards a framework of player experience research. In *Proceedings of the 2011 foundations of digital games conference, EPEX 11*. Bordeaux, France.
- Pagulayan, R., Keeker, K., Wixon, D., Romero, R. L., & Fuller, T. (2003). User centered design in games. In *The human-computer interaction handbook: Fundamentals, evolving technologies, and emerging applications* (pp. 883–903). Mahwah: L. Erlbaum Associates.
- Rud, O. (2009). *Business intelligence success factors: Tools for aligning your business in the global economy*. Hoboken: Wiley. ISBN 978-0-470-39240-9.
- Salen, K., & Zimmerman, E. (2003). *Rules of play. Game design fundamentals*. Cambridge, MA: MIT Press.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine*, 15(9).
- Tufte, E. (1983). *The visual display of quantitative information*. Cheshire: Graphics Press.
- Watson, H. J., & Wixon, B. H. (2007). The current state of business intelligence. *Computer*, 40(9), 96. doi:10.1109/MC.2007.331.

- Web Analytics Association. (2007, August 16). Web analytics definitions. URL: http://www.webanalyticsassociation.org/resource/resmgr/PDF_standards/WebAnalyticsDefinitionsVol1.pdf
- Weber BG, John M, Mateas M, Jhala A (2011) Modeling player retention in Madden NFL 11. Innovative Applications of Artificial Intelligence (IAAI). AAAI Press, San Francisco
- Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques*. The Morgan Kaufmann Series in Data Management Systems (3rd ed.). Morgan Kaufmann.

Chapter 3

Benefits of Game Analytics: Stakeholders, Contexts and Domains

Alessandro Canossa, Magy Seif El-Nasr, and Anders Drachen

Take Away Points:

1. It is vital to discuss the varied backgrounds and epistemological biases for each of the game development disciplines with stakes in game analytics.
2. Understanding the different expectations, potential benefits and requirements that each type of stakeholder brings to the table when discussing game data analysis is required prior to establishing an efficient analytical strategy.
3. A number of concrete examples are provided for each stakeholder involved in an ideal analytical strategy.

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

M. Seif El-Nasr, Ph.D.
PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

A. Drachen, Ph.D.
PLAIT Lab, Northeastern University, Boston, MA, USA
Department of Communication and Psychology, Aalborg University, Aalborg, Denmark
Game Analytics, Copenhagen, Denmark
e-mail: andersdrachen@gmail.com

3.1 Introduction

The process of creating a game is a perfect example of a multidisciplinary effort. Development teams are extremely heterogeneous, involving writers, programmers, artists, designers, engineers and architects, just to name a few. Each discipline brings a legacy of assumptions, practices and values that, more often than not, are far from aligned with each other. For example, designers often discuss with programmers in order to modify existing features or introduce new ones based on their past experiences, theoretical assumptions or “gut feelings”. User researchers often act as mediators in these transactions, and starting these discussions based on objective evidence and telemetry data has proven beneficial. It is, therefore, necessary to understand and chart the different possible benefits, biases, angles, needs, costs and requirements that each discipline brings to the table when it comes to game data analysis, since a thorough understanding of these facets can considerably help in defining a strategy to select interesting game variables to monitor.

In this chapter, the principal stakeholders who might have an interest in game analytics are listed and examined in light of their background, the expectations and requirements and potential benefits that game analytics can provide and finally, where relevant, examples of game analytics are given based on the content of this book.

- **Background:** who are the stakeholders, what are their main tasks and goals.
- **Expectations and requirements:** the underlying needs of different stakeholders, the limitations in terms of output and input: what data, information and format a certain stakeholder requires before starting any analytics and the outcome that each stakeholder expects from game analytics practices.
- **Benefits:** how analytics can improve the performance of different stakeholders.
- **Example:** a case study that will be explored in detail in this book.

3.2 Game Designer, Level Designer, System Designer, Game Director

Background: Designers are probably the most heterogeneous group of stakeholders; they have a wide variety of possible backgrounds from architecture to film studies to computer science. But they share one important trait: they are all responsible for creating the design and mechanics for the game.

When dealing with game telemetry and analytics, designers often generate very specific hypotheses that need confirmation or falsification to fine tune and balance elements of the game from weapon damage to navigation flows. In these cases it is fairly straightforward to pinpoint the relevant game variables, transform them into metrics and extract features. Often game directors and lead game designers are also interested in exploratory analyses aimed at creating models of player behavior.

Addressing a request for precise answers, Microsoft Game Labs pioneered telemetry-based systems to support user testing of Halo 3, generating metrics-based

analyses of player progression and heatmaps to answer concrete questions from designers (Thompson 2007; Kim et al. 2008).

Until recently it may have been slightly challenging to convince designers to make use of analytic intelligence to corroborate their decisions, these days more and more designers have learned to appreciate the edge given by precise information; an important challenge for designers in the future could be to avoid growing completely dependent from analytics practices but to maintain a certain trust their in own vision.

Expectations and requirements: designers often expect to use analytics to improve the design of the game systems they are developing, specifically in terms of gameplay, navigation and balancing issues. Analytics are required to provide designers with low-level and detailed information about player behavior, such as interaction with mechanics and other players. Another important application that transcends a single game is tracking players' preferences in each installment of a franchise for long term development of that franchise.

Benefits: to designers, analytics provide a great way of fine-tuning player experience, provided they have a solid initial vision of what experience the game should elicit.

Examples: in Chap. 14 a plethora of hybrid techniques are utilized by designers to gather insights, in Sect. 14.3, a case study is presented that shows how designers leveraged analytic intelligence to improve the distribution of ammunitions for the game Kane & Lynch 2: Dog Days (Square Enix, 2010). Additionally, in Sect. 14.4, a further case study showcases how analyzing the different cause of death allowed phenomenological debugging for Fragile Alliance 2, the multiplayer experience of Kane & Lynch 2. Another analysis of causes of death in Tomb Raider: Underworld (Eidos Interactive, 2008) is presented in Sect. 14.5. Chapters 18 and 19 showcase several examples of design intelligence gathered through visualizations, Chap. 17 focuses on spatial analysis, while Chap. 12 demonstrates how data mining can be used by designers to assist designers in balancing the game and discovering game design issues.

3.3 Producers and Project Managers

Background: producers and project managers are concerned with managing people involved in development and assessing people's performances and production schedules; it is therefore necessary for producers to have very precise estimates for all sorts of tasks. Beside individual estimates given by each team member for each task, producers and project managers also take advantage of automated tracking of processes and pipelines: the metrics of interest are mostly related to turnaround times for new content, barriers and slowdowns on the development pipeline, time for each production iteration, etc. (Mellon 2009). However, defining which metrics to collect in order to monitor production processes is not straightforward since

evaluating people's performance often involves qualitative assessments; the problem is amplified when dealing with artwork since it is very difficult to create objective metrics that take into account art directors' evaluations of different assets and their level of polish and quality.

A secondary area of interest for producers is the relation between production costs and player use: any feature, mechanic, system, character or asset in the game has a cost; its acceptability is relative to its centrality to the core of the game, its visibility/accessibility and its use by players. So producers can easily assess a game element in terms of its cost in man/hours and how used/visible it is.

Expectations and requirements: producers expect to use analytics to maintain better overview and control of production schedules. Executives want high-level summaries on production status and game feature use, in order to better control the scope of production.

Benefits: for these stakeholders, analytics provide a speedy, more informed way allowing them to make production decisions regarding personnel and resource allocation.

Example: Chapter 15 contain an interview with producers Aki Jarvinen (Digital Chocolate) shedding light on how management can better maintain control and set goals for production of digital games, directing where to focus the team efforts and how to distribute resources.

3.4 Marketing Managers

Background: Marketing managers are mostly interested in monetization, leveraging both traditional and alternative revenue streams, such as micropayments, pay per play, freemiums, etc. They are specifically interested in telemetry that tells them: who buys what, how often, where and when. The most notable efforts in this direction were spearheaded by Zynga, where dashboards are included in several games allowing rapid A/B testing, leading to marketing-driven development models (Pincus and Gordon 2009). Other developers and publishers, such as EA, are investigating ways to monetize gameplay analytics and intelligence and are actively researching into player retention for their sport franchises (Weber et al. 2011).

Expectations and requirements: marketing managers expect to use analytics to increase game and micro-transaction sales and they require an analytics system that provides instantly updated information about every in-game or in-store transaction.

Benefits: for marketing managers, analytics is valuable as it can pinpoint piracy very early and act rapidly to prevent it. Furthermore, it can provide an insight towards which game items and features are more lucrative. This intelligence is widely used to plan new content according to specific demographics and psychographics.

Example: Chapter 5 presents an in-depth discussion of Zynga's practices related to analytics and marketing, while in Chap. 24 Simon Møller of Kiloo discusses a

co-production model that aims at capitalizing on experiences and strategies from game to game to maximize revenue streams for the mobile market.

3.5 User Experience Research

Background: Traditionally user research departments have adopted qualitative approaches as part of their practices given how they are mostly concerned with subjective, soft concepts, such as player experience, satisfaction and engagement. Therefore, phenomenological debugging is carried out through interviews, think-aloud sessions and thorough observation. Recently there have been several success stories of qualitative and quantitative triangulations: where game data analysis aided user researchers in pinpointing soft, subjective states, such as frustration (Sørensen and Canossa 2012). Game telemetry and analytics are starting to appear as the perfect companion to other existing methods, such as usability testing (measuring ease of operation of the game) and playability testing (exploring if players have a good experience playing the game), to offer insights into how people are actually playing the games under examination (Thompson 2007; Kim et al. 2008).

Expectations and requirements: game user researchers expect to use analytics to help augment the current user research methods and to give researchers a clear and detailed indication of what the player is doing at any given time. Researchers require analytics systems that can provide them quickly and easily with detailed information to aid them in their investigations for the perfect player experience. Large-scale datamining provides information after a game has shipped; hence, purely quantitative methods are applicable only to games that can be easily modified such as web games and MMOGs. User experience research for traditional console titles requires a fast turnaround fitting the various iterations and stages of the production cycle. In fact, only a handful of studios can afford the massive beta tests with thousands of players that are necessary to gather statistically significant datasets. Most developers rely on a much more limited number of testers, between few dozens and few hundreds. These numbers are rarely sufficient to justify heavily quantitative analyses. Furthermore, qualitative methods have proven successful in gauging subjective emotional states such as elation, frustration, fear, etc. It is therefore necessary that quantitative game analytics blends smoothly with existing practices employed in user research departments.

Benefits: for game user researchers, analytics provide very clear and concise clues which can be used to show what players are doing; these clues can support the conclusions that researchers draw about problems in the game. Even though interviews and other methods provide valuable insights, it is often beneficial for user researchers to show numbers to other stakeholders in order to prove a point or to easily convince designers of the necessity of the changes suggested. Furthermore, several examples in the book show how triangulating qualitative and quantitative methods can provide answers to questions that could not otherwise be asked.

Example: Chapter 12 showcases several examples of a user research manager cooperating with academics to address specific questions asked by designers for a

number of different Square Enix games. Chapters 21 and 22 discuss several examples of game user researchers working on triangulating telemetry with other measures and the discussed benefits and successes.

3.6 Community Managers

Background: community managers are in charge of leveraging the community of players around a certain game. Usually they are dedicated fans that acquired professional formation as communication experts and have extensive knowledge on how to leverage diverse communication channels to benefit a certain game, brand or franchise. They keep under control the social connections between players, organize events and challenges, try to maintain high level of interest in a game and sometimes handle customer support requests.

Expectations and requirements: community managers expect information regarding top performers, opinion makers, play styles, achievements and social relations between players. This information is crucial to grow brand awareness and maximize game lifecycle. This type of stakeholder is usually concerned with high level, highly aggregated and abstract statistics that can give an insight on how to increase player retention (Weber et al. 2011). Social networking features have also become key areas for viral communications, while forums have provided for years valuable intelligence about the community's attitudes.

Benefits: Game telemetry has been widely used by game studios to build and maintain communities. For example, Valve was among the first to make available to their community highly aggregated data about player behavior (Steam Games Statistics). Bungie took it a step further by not only providing aggregate statistics about large number of players, but also presenting each player with personalized reports (Bungie Games Statistics). Community managers can harness this information to create *ad hoc* challenges, promote and reward certain behaviors and maintain overview of the community's attitude towards a game. Often, benefits to community managers translate directly into benefits for players, as seen by the rapid diffusion player's dossiers.

Example: Sections 18.3.2 and 18.3.3 presents interesting case studies of visualizations produced by third-parties or players that can be utilized by community managers to explore and exploit social networks.

3.7 Third Party Data and Analytics Providers

Background: Outside the companies that actually develop computer games and other forms of interactive entertainment, a rapidly increasing number of third-parties have emerged in the past few years to provide analytics-related services to

companies. These form two broad categories: consultants and middleware providers. The first category specialize in advising companies on how topics such as how to develop an in-house analytics solution, training of staff, advice on what aspects of performance, processes and users to track and how to analyze the resulting game intelligence (see Chap. 13), and other topics that a company might have use for. The middleware providers supply the actual software and sometimes infrastructure needed to build an in-house analytics capacity, this commonly through one of four channels: (1) The provider constructs an in-house infrastructure and supplies the necessary software; (2) The providers deliver a software package that permits the game developer to collect data, analyze them and develop reports. Usually the developer licenses the software. The vast majority of middleware providers in this category are focused on user telemetry, including typically in-game behavior, purchase records, user accounts, and sometimes more exotic data sources such as forum post tracking. The coverage and quality of analytics packages currently on the market varies substantially. Examples include Tableau and other business intelligence packages. (3) The providers deliver software-as-service (SaS), in which case the game developer uses the middleware system to design hooks in the game code, which transmits player behavior telemetry to collection servers, from which the developers can interact with the data via an accompanying UI. This is one of the currently most popular business models among small-medium sized developers. Usually payment is scaled according to traffic, e.g. number of messages. As with the licensed software option, the input data sources accepted varies from product to product, with current SaS-companies generally focusing on Free-to-Play (F2P) games, i.e. supporting the tracking of player behavior and user purchases, as well as allowing for the importing of fourth-party data, for example click-streams or user account information, in order to permit funnel analysis, marketing channel analysis and similar (see Chaps. 4 and 12). Some companies provide an option to choose between SaS or licensed solutions. At the time of writing there are relatively few middleware providers solely dedicated to games. Examples of middleware providers include HoneyTracks, Game Analytics, Games Analytics and Playtomic. (4) An entirely different third-party option, commonly used by players, are performance tools such as P-stats, which provide a small application that is installed on the local client, which transmits personal play behavior data to a database, which the player can interact with via a web-based interface. Usually these solutions are open, i.e. everyone can see the statistics of all members of the network, and thus forms excellent tools for e.g. Battlefield clans to keep track of member performance.

Typically the people behind third-party companies are professionals with experience from large-scale data storage, database construction, programming, game data mining, game analytics, information visualization, game design and game development.

Expectations and requirements: Middleware providers have no expectations or requirements from the data feeding analytics, being usually focused on delivering data to the game industry instead. However, third-party consultants will have requirements that match the analysis they are expected to run. For example, if asked to analyze the in-game economy of a massively multi-player online game, the

consultants will need telemetry on various aspects of the flow and aggregation of financial resources in the game world system. Usually, if third-party providers are granted more access to data this results in deeper, more comprehensive analyses.

Benefits: The immediate benefit to a game company for using third-party providers to help with game analytics – as with in any other consulting/middleware solution – is that they avoid the requirement for building expertise and/or tools internally, saving considerable resources. How much and what depends on the type of solution chosen.

Example: While there are no chapters dedicated exclusively to third party providers of analytics middleware or consultancy companies in this book, Chap. 10 describes a basic RestAPI-based telemetry system, and Chaps. 12 and 17 discuss in brief the role of third-party middleware providers for supplying the logging, transformation, storage and access to telemetry data, as well as varied degrees of analysis and visualization. Darius Kazemi, interviewed in Chap. 11, was President of Orbus Gameworks, one of the earliest game analytics companies.

3.8 Programmers

Background: Different programmers have different concerns. For example, engine programmers might be interested in frame rate, memory loads or compiling errors, while network programmers are more concerned with server stability and hardware performance. A common trait for all the different metrics needed by programmers is that they belong to the class of performance metrics, which are rather straightforward to collect and analyze but require a communal effort. Concerns regarding the complexity and cleanliness of the code are more difficult to address since it is often a subjective measure and, as such, harder to quantify. Mellon described the process of collecting performance data from automated testing sessions in *The Sims Online* (Mellon 2009).

Expectations and requirements: If hardware and software performance metrics, such as server performance, data archiving and bug tracking, are represented in orderly, logical manner, understandable by anybody in the team, then it is possible for everybody to have constant access to the specific game build status. Programmers often require detailed information about the symptoms of a runtime or compile time error that has occurred in order to diagnose the problem and fix it. Thus, they would require tools and methods to show details of actions that led to the specific errors. Telemetry can be used to give a clear indication of actions performed prior to the error event. Visualizations, such as heat maps can show where in the level crashes have occurred. Other programmers are more interested in efficiency issues, and thus benchmarks are important as well as a picture of what the context is like when specific delays happened.

Benefits: a coordinated effort allows programmers to tackle eventual problems as a team and not as individuals; enhanced coordination results in faster operating time.

Example: In Sect. 18.3.1 it is shown a system that clearly addresses the programmers' need to maintain overview and coordinate efforts during bug tracking. Chapters 6 and 7 also dwell further on these cases.

3.9 QA Manager and Testers

Background: This category group of users are concerned with the number and severity of bugs: the benefit of integrating metrics tracking systems with existing bug tracking systems is that it would help in figuring out how to replicate a bug if all game variables were tracked prior to the manifestation of the bug. Only in this case it would be recommended to indiscriminately monitor all features, and only because no analysis has to be performed on the dataset, just a review of the metrics events that preceded the manifestation of the bug.

Expectations and requirements: testers often require knowledge about bugs: how and when they occur. Thus, they would require tools and methods to show details of the bug that occurred and the actions that led to the specific errors that can be relayed to programmers. QA managers and programmers have similar requirements in terms of bug tracking, but programmers expect more aggregated reports from QA managers.

Benefits: comprehensive and automated tracking of in-house QA testers greatly increases efficiency

Example: Chapter 7 presents more information on these cases with a case study based on SkyNet, a system developed at Electronic Arts.

3.10 Third Party Services for Players

Background: third party services focus on aggregating data from several games with the purpose of providing added value to end users, players at large, by offering recommendations, matchmaking or social networking features not for a single game but for a large number of titles developed and published by companies that, more often than not, are in competition with each other. These services analyze game data to understand player behavior and connect players. The challenge is then to homologate often inconsistent data from very different datasets, for example Metacritic (2001) is forced to normalize game scores from a number of reviews that sometimes do not even provide a numerical value.

Expectations and requirements: leveraging the capability of bringing together information from different sources, these services can provide players with precious information about the games they play, their own play experiences across different games or compare their experience with their peers. Considering how

dis-homogeneous the datasets can be, it is required to enforce some form of consistency within each dataset, whether it is review scores, achievements or raw data such as play time.

Benefits: utilizing these services players have access to a wealth of information not always provided by developers, across several games ranging from summaries of stories, characters and mechanics, to scores, achievements, progression, on-line presence of friends and detailed data about their – and their friends’ – purchase history or play experience such as time spent with each game.

Example: Section 18.3.2 provides several examples of third party services.

3.11 Players

Background: players are the end users of the game artifacts. Developers devote considerable resources to provide players with analytic tools in the form of statistic tracking or player dossiers, and they keep improving them, for example the WOW armory (reference) has been through a number of iterations. Yet, dedicated players, using APIs provided by developers, often build their own tools for exploring game data either because they want to expand on the limitation of the tools provided by the developers or because there are no tools available. An example of the first case is seen with replays of *Starcraft 2* and the player-built tool *SC2Gears*, while an example of the second case is *Minecraft* and *Minecraft X-Ray* (see Chap. 18).

Expectations and requirements: players desire to use game data to be informed about their performances, and to augment their gameplay experience and would greatly value easily accessible APIs and developers’ support.

Benefits: by creating additional tools, players project their own cultural habits, modifications and ideas on to their gameplay

Example: Section 18.3.3 provides several examples.

3.12 Conclusions

These are just a few of the types of stakeholders that collectively contribute or affect the development of a game. There are many more, and although they do not figure in this book, they will take an active role in shaping the discipline of game analytics; for example localization producers already expressed interest in understanding exactly which parts of the game were used by foreign language testers, even writers, concerned with the clarity and intelligibility of their texts, are making active use of player metrics – which dialog choices are the most popular – to revise different drafts of their lines.

As we have seen until now, each group of stakeholders requires a different feature set. Some require data to be collected during production, others after launch and others again during the whole lifecycle of the game. As we will see in Chaps. 7 and 18, multimodal, dynamic reporting and visualization systems seem to have been able to address the varied nature of expectations and requirements set forth by all stakeholders on the development side. In fact, drill-down reports can supply both high-level information to executives and at the same time generate detailed reports for team leads from design to programming; eventually each single developer has the opportunity to generate individual reports with very different foci: for example, mapping the impact on performance of a certain shader, or the player interaction with a certain system or the navigation flow on a certain portion of the level. No matter the efforts that developers devote to provide the community with tools to explore their play data, there will always be space for both third party services and player-created tools to bridge the irreconcilable gap of information that competition between different publishers inevitably leaves open.

About the Authors

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and the Royal Danish Academy of Fine Arts, Schools of Architecture, Design and Conservation. His doctoral research was carried out in collaboration with IO Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches. His work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he maintains an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio.

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. Magy worked collaboratively with Electronic Arts, Bardel Entertainment, and Pixel Ante.

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated

with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

References

- Kim, J. H., Gunn, D. V., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution conference on human factors in computing systems, CHI'08.
- Mellon, L. (2009). Applying metrics driven development to MMO costs and risks. White paper, Versant Corporation.
- Pincus, M., & Gordon, B. (2009). Ghetto testing and minimum viable products. Lecture at Stanford Technology Ventures Program.
- Sørensen, J. R. M., & Canossa, A. (2012, March). Arrggh!!! – Blending quantitative and qualitative methods to detect player frustration. Game developers conference, San Francisco.
- Thompson, C. (2007, August 21). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine*, Issue 15.09.
- Weber, B. G., John, M., Mateas, M., & Jhala, A. (2011, August). Modeling player retention in Madden NFL 11. Innovative Applications of Artificial Intelligence (IAAI), San Francisco, CA: AAAI Press.

Chapter 4

Game Industry Metrics Terminology and Analytics Case Study

Timothy Victor Fields

Take Away Points:

1. Online games benefit from tracking information about gamer behavior.
2. The practice of preparing software to collect this data is called instrumentation.
3. The data gathered is referred to as metrics.
4. These metrics can be used to compare the performance of one game to another.
5. This chapter describes and defines commonly used metrics.
6. The metrics gathered can provide valuable insight into how developers can improve their game.

4.1 Introduction: Industry Terms and Metrics

The advent of online gaming has made it possible for almost every game device to send information across a network. The evolution of consumer tastes has forced the variety and quality of marketplace offerings to expand. Market competition and the consumer expectation that games can be constantly improved through the issue of online patches have forced game developers to stop thinking of their games as ever truly finished. Instead, mobile, console, PC, and social games are all expected to monitor user behavior, and report information back to the game creators. The developers responsible for monitoring and improving games are then able to use the information the client software sends to determine the sorts of improvements they should make to the game. These improvements are rolled out as patches or updates to the client software, or changes to the runtime code or content on the game servers.

T.V. Fields (✉)
Capcom Vancouver, 3993 Nithsdale St, Burnaby, BC V5G1P5, Canada
e-mail: tfields24@gmail.com

The key to creating a successful automatic feedback loop between the customer and the game's creator relies on measuring how the user interacts with the game. This information is collected via instrumentation in the software, which sends telemetry data back to a server. These specific bits of information are commonly referred to as "game metrics."

Modern game developers use a variety of terminology for the various statistics they measure. Much of this terminology is borrowed from the online advertising space, and very few standard definitions are agreed upon. As many of these terms are abbreviated, newcomers to the industry may find themselves faced with a daunting array of acronyms covering user numbers and monetization. Since each game is a little different, each developer values slightly different key metrics, but invariably a few central metrics end up being the most important. Most online games put a strong focus on *ARPU* (Average Revenue Per User) or *DAU* (Daily Average Users) as ways of measuring user engagement. The introductory chapters of this book gave readers an overview of such metrics and their use within the game industry. In this chapter, however, I am going to discuss in more detail a much broader array of potential statistics, and the logic behind their utility.

The simple truth is: *metrics matter*. Anyone who cares about the performance of a game needs to learn to understand the study of user data. Regardless of the platform, users no longer expect "fire-and-forget" games; modern games on PC, console, and mobile devices must analyze user feedback and experience and use this information to adjust their offerings, thus making themselves appealing to their customers. In a nutshell, this process is an analogue to the scientific method. A developer forms a hypothesis about user behavior, gathers data to test the hypothesis, and then makes a change to the game. The team then releases a patch or updated version of the product and runs the "experiment" again, gathers the metric data, and analyses it to see if their hypothesis was accurate, and if the alteration to the game had the desired result.

With a very few exceptions, games are produced as part of a business enterprise; game companies seek to profit from their games' release. As traditional retail models have fragmented into dozens of competing types of digital distribution, more and more games have adopted a free-to-play business model.

This term is something of a misnomer, since these games are designed to encourage the user to pay money in small amounts (called micro-transactions) in order to enhance their experience; thus, many users playing them end up paying money to improve their experience. Since the user's choice to pay is entirely optional, game designers who seek to profit from free-to-play games are forced to carefully study game metrics in order to determine how to tweak the game design in order to maximize per user spending, while still providing a satisfying experience.

While each game differs, and will need to design an instrumentation and metric gathering plan tailored to its specifics, the individual nuances still usually rest on a common framework. This chapter will help explain common user metrics, why they matter, and how the different types of games can benefit from analysis of such data. I will mostly focus on metrics commonly used in online MMO, free-to-play, and social games (which have tended to be pioneers in the use of telemetry data).

However, most of these metrics are broadly applicable for games in general, as most games moving forward have some online component.

In this chapter, I will first start by discussing the general terms used within the industry when referring to metrics and analytics (Sect. 4.2). I will then discuss the metrics that are most widely used (Sects. 4.3, 4.4, 4.5 and 4.6). Following this discussion I will then outline a case study to materialize the metrics discussion (Sect. 4.8).

4.2 The Business of Metrics

Before we dive into the meat of metric analysis itself, I will define some of the common high level terminology used to describe the art, business, and science of gathering data from games.

4.2.1 *Business Intelligence*

The *Business Intelligence* unit is often used to refer to the analytics department of a software publisher, media, or games company. *Business Intelligence* typically is involved the sorts of topics discussed above, from determining what type of information to gather, to more rigorous data processing and analytics efforts that are involved in crunching numbers and confirming statistical validity. Finally, the role of *Business Intelligence* at most companies is to help interpret the data derived from metric gathering and turn it into actionable strategies for the company or products moving forward.

4.2.2 *Analytics*

Analytics refers to the science of gathering information from the runtime user interactions with a piece of software or website and processing it. *Analytics* is a discipline that combines statistics, engineering, and software design.

4.2.3 *Metrics*

The term *Metrics* refers to the particular bits of data that the client software reports back to the server in order to help developers and the *Business Intelligence* unit make sense of how users are interacting with the game. *Metrics* is also used to refer to the

measurable criteria for success that developers are trying to reach, e.g., “Our target *metrics* are 98% server uptime and a user engagement of 3 hours per day.”

4.2.4 Instrumentation

Instrumentation is the process of putting functions into a piece of software to gather and report back *Metrics* to the *Business Intelligence* department. Most modern companies instrument their games to collect *metrics*. It is important to note, however, that there are third party middleware solutions that can ease this process. The individual components that measure various elements are sometimes referred to as “hooks”; e.g., “We are putting hooks into the new version that will let us know how many users finish the first level.”

4.3 Measuring Player Population

No matter else what happens, without users, the game is a dud. Even if they don’t monetize initially, a huge number of users can be the golden ticket to a successful product. For example, consider Facebook or Instagram, who spent years simply “capturing eyeballs” with no clear plan to create revenue. In this section, I discuss some key metrics that are currently used within the industry to evaluate the overall population of a game.

4.3.1 DAU – Daily Active Users

DAU is a measure of the number of unique users per day, usually calculated over the last 7 days. Typically, there is no minimum play time or further interaction required to qualify as a “daily user.” This makes *DAU* a fairly shallow metric, in that it ignores the concept of user engagement, except by measuring population in the aggregate.

Most social networks, for example, consider a user active when the user “views or engages with your application or your application’s content.”

Actions that count towards *DAU* on Facebook games:

- Gamers who visit the game’s splash page
- Gamers who publish items to their news feed from a game
- Users who “Like” another user’s post from a game
- Users who commented on a post made by the game

All but the first of these items are largely irrelevant outside of social network games. In fact, many seem to exist as a way of allowing social networks to “add value” to a game by artificially inflating the *DAU*. For games that are not on a social network simply sending out a bit of data to the server when a user fires up the game is usually sufficient to capture them as part of a *DAU* metric.

As we can see, many of these are extremely shallow interactions, and not particularly meaningful in determining how to better engage, monetize, or delight users. Still though, keeping track of how many users play your game on a given day is an excellent, if gross, way to track and identify broad trends, which can speak to the game’s overall popularity. Furthermore, refining this information by indexing users per geographic territory, or determining which users are playing on a particular device can be quite useful.

4.3.2 MAU – Monthly Active Users

Monthly Active Users (also often referred to as *Monthly Average Users*) is a measure of the number of users in a given calendar month, typically calculated from the first to the last day of that month. Occasionally *MAU* is also shown as a current floating 30 day average, particularly on social networks like Facebook. In effect, *MAU* is an aggregate of the *Daily Active Users (DAU)* over a month.

For research purposes, developers and students should clarify if they are attempting to track “unique” users or not. In many cases *MAU* numbers are drastically inflated by users who play a game multiple times per day or month. To count unique users, each user would only be counted once towards the total daily or monthly count. For example, if we were only tracking unique users, and 1,000 people played a game and each of them logged in once per day, every day of the month, *DAU* and *MAU* values would be identical. But if tracking unique users was unimportant, and every user logged in every day, the numbers could show 1,000 *DAU* and 30,000 *MAU*. Often the difference between these measures is abbreviated as *MAU* and *MAUU* (Monthly Average Unique Users). Both can be useful.

Developers often refer to a game as being “sticky,” meaning that users who try it out stick with it and keep returning for more. You can use the relationship between *DAU* and *MAU* draw some conclusions about the “stickiness” of a game.

By calculating *DAU/MAU*, we are able to get a measurement for how many players show up to play each day. Imagine you gather data which informs you that a particular game has one million *Daily Active Users*, and three million *Monthly Active Users*. Such a game has a *DAU/MAU* ratio of .3 or 30%; a third of that game’s players are registering least once per day. This number measures how “addictive” a game is for its users. The higher the ratio, the “stickier” the game, and the more successful it is at engaging users. Most games consider it a great success to become a part of their users daily routine; this keeps users thinking about the brand and spending money on the product.

One common use to which this type of information can be put is when measuring the effect of game design changes on your user base. If a developer changes the way the user interacts with the game – something crucial in the first few minutes, or something that affects the core gameplay loop – they can study how their *DAU/MAU* changes. If the ratio improves, the change is probably a good one. Of course, given the number of variables associated with so crude a measure, it is often difficult to pin down just one thing that changed and even more difficult to isolate all the other factors that may have been responsible without correlating many other pieces of data. For example, a falling *DAU* between a Sunday and a Monday might have nothing at all to do with a change in the design of the software. Most often filtering your analysis of new data based on historical trends is a critical part of correctly evaluating metrics.

This *DAU/MAU* ratio can be used as a high level indicator of stickiness. If users keep returning to a game, then the first and hardest part of making the game a success has been accomplished. At that point, the developers need only increase the total volume of users, usually through advertising, and focus on increasing the percentage of users who monetize in order to make the game a financial success. This ratio seems to serve as an accurate predictor of success for games of all sizes, from 20 million to games with only 4,000 players. If the game is sticky, then the core design is probably a success.

When the *DAU/MAU* ratio is low, game developers should focus on improving the core game rather than spending much to market the game. Why? Because a low *DAU/MAU* is a good indicator that users aren't engaged enough by the game to come back. This problem is less important for retail games or for "premium" games, where the user must spend money to purchase the software before trying it than it is for "freemium" games, which let users play for free in the hopes of making their money on micro transactions. This popular free-to-play model only works if the first taste the users get is so compelling that they regularly come back for more. Until this ratio looks good for a thousand users, there is little value in advertising to pull in a million, because it is unlikely that most will return to keep playing and paying.

4.3.3 PCU – Peak Concurrent Users

For games with a strong backend server component, like MMOs server demands have historically been a significant cost. This makes it important for those who plan the backend support needs for a game to understand how many users are playing the game at any given moment. This notion is referred to as "concurrency" and the high water mark for this number is referred to as the game's *Peak Concurrent Users (PCU)*. Poor planning for the *PCU* a game might experience can result in unacceptably long wait times as users queue up for server time, or worse, server crashes. Understanding a game's likely *PCU* can also be used to make decisions about how many customer service people are required at peak times, and other planning details, which affect operational costs.

In the last few years the advent and rise of rapidly scalable cloud computing, which allocates server power based on real-time analysis of demand, has somewhat reduced the importance of this type of statistic. In traditional client-server models, the game developer was forced to maintain the physical server hardware to handle logins, game simulation, etc. If the game ended up being more successful than expected, the strains on this server hardware often led to poor user experiences, long wait times to login, or similar. With newer models, in which the server side software is run by large server hosting companies, like Amazon or Google, this computing is done “in the cloud.” As concurrent user numbers rise, the cloud service provider can automatically scale up the computing resources allocated to the game.

4.4 Measuring Monetization

There are a variety of ways to measure how well a game monetizes its user base. The most important thing to remember is that if a game uses a typical “freemium” model, even very high *DAU* numbers do not in any way signify financial success for the game unless these players are converted into paying customers. The only way to understand a free-to-play game’s revenue stream is to analyze how many users the game has, how many of its users actually pay, (usually expressed as a percentage), and how much each user pays on average over the course of their engagement with the game. This section covers the common analytics used to measure and estimate these values. These metrics are of particular importance in financial forecasting within the entertainment software industry.

4.4.1 Conversion Rate

When users try out a free-to-play game, or a game demo, how many are converted into paying customers? This is a critical measurement for any free-to-play game. The particular monetizing model can vary greatly, from new pinball tables in Xbox Live Arcade’s *Pinball FX2* (Microsoft Game Studios, 2010), to new summoner runes in *League of Legends* (Riot Games, 2009), to more energy or turns in *Empires and Allies* (Zynga, 2011). However, the core measure of what percentage of a game’s total users ends up paying is a critical metric.

4.4.2 ARPU – Average Revenue Per User

Average Revenue Per User is a measure of the total revenue for a given time period (say, a month) divided by the total number of users that month. The resulting value is a measure of average revenue per user. This number can inform planning teams how much revenue they can expect, on average, for every customer who plays their game.

Beyond the sheer glee of counting future profits, a study of *ARPU* can be extremely valuable when trying to determine how much money to spend to attract new users. In general, if the amount required to attract a user, and the amount required to provide service to a user once they exist are less than the *ARPU*, then a game can be scaled up through advertising to increase its profitability.

4.4.3 *ARPPU – Average Revenue Per Paying User*

One of the sad truths of the free-to-play model is that most users never pay anything for games, which are free to play. Luckily, those who do tend to pay, pay enough to make up for all of those who choose not to engage with any premium content. For this reason, thinking of revenue across all users can obscure a key truth: some users are worth far more than others. To express this nuance, developers think about Paying Users, and the resulting value of *ARPPU*.

To calculate this value, developers take the total revenue for a given time period and divide it up amongst the total number of users who converted during that same time period. For subscription-based games, like Blizzard's venerable *World of Warcraft* (Activision, 2004), where all users have a fixed subscription price, this amount is quite easy to predict and is virtually identical to the *ARPU*. But for free-to-play games, the *ARPPU* is usually much higher than *ARPU*, since most users do not pay anything, and this metric only tracks the average for those who do.

4.4.4 *UAC – User Acquisition Cost*

What does it cost to attract a new user? In order to get users, game developers must first invest in building the game, test it, get the game authorized to be distributed on its target platform (through Sony, Apple, Microsoft, Facebook, etc.). Then they need to get the message out that the game exists, usually through advertising or manipulating various viral channels on social networks. The aggregate of all of these costs divided by the total number of users represents the *User Acquisition Cost*. Due to the high cost of product development, the first user is by far the most expensive; the second user costs half as much to acquire, and so on. Eventually these costs level out such that the cost of each additional user can be easily calculated.

Understanding these costs can help inform advertising strategy and budgets. When evaluated against *ARPU* developers can determine if it makes sense to spend to acquire new users. Effective use of viral channels, "word-of-mouth," and prominent placement on platform storefronts, such as the iTunes Store or the PSN dashboard, can further reduce these costs. It is a truism of social media that the cost to acquire a new user declines rapidly initially, then begins to increase once awareness of the product reaches a saturation point among those most likely to be customers. Thus, *UAC* increases when trying to expand a product offering into a new demographic or previously under-served market of customers.

4.4.5 *LTV – Life Time Value*

No game has yet been in operation long enough to hook users from the cradle to the grave (though some of the more popular MUDs and MMOs are getting close). Instead, most games have a limit to how long they can engage a particular user before that customer quits playing and moves on to enjoy some other diversion. When a user stops playing, they almost always stop paying, and stop telling their friends about the game. This means that, effectively, the user who stops playing no longer has any measurable value to the game. *Life Time Value* studies the total amount of money a user will contribute to a game's bottom line while they are engaged. For retail products or "premium" digital games with no additional monetization methods, *LTV* is equal to the initial purchase price of the game. Subscription model games monetize a user each month for as long as they are engaged, and so on. For micro-transaction models, *ARPU* multiplied by the number of time units that comprise a user's life cycle with the game results in *LTV*.

By way of example, if an average user buys \$2 worth of virtual goods each month, and on average every user remains engaged in playing the game for 24 months before they quit, then that game could describe an *LTV* of \$48 per user. The relationship between *LTV* and *UAC* can help determine each user's value as profit to a company.

4.4.6 *LTNV – Life Time Network Value*

It's a shortsighted merchant who thinks of a customer as no more than the contents of their wallet. Users can add far more value than their mere dollars. Vocal supporters of a game help to spread it, socially active "mavens" help increase virality, and active multiplayer communities can nurture new users. Moreover, in today's marketplace, where much of sales is about making it onto a "Top Downloaded" list, crowds begat crowds.

All of these types of non-monetary value, added to the *LTV* of a user is their *LTNV*. This value is difficult to calculate, but important to understand in order to plan effective strategy around how to support and market a game.

4.5 Online Advertising

4.5.1 *CTR – Click Through Rate*

Online advertising, from banner ads to popups to console dashboard placement is seen by many more people than those who chose to interact with it. These views are called *Impressions*. The percentage of those who choose to engage divided by the

number of people who see an ad is referred to as its *CTR*. This value measures how successful an ad is at converting awareness to interest. When a user sees an ad, they are now aware of the product. If they click on it, they have expressed interest. By way of example, if 200,000 people see an advertisement, and 800 of them click on it then the ad has a *CTR* of .4%.

4.5.2 CVR – Conversion Rate

The percentage of users who progress from looking at an ad or using a free trial to becoming paying customers is referred to as the *Conversion Rate*.

Conversion rates are usually expressed as a percentage. When describing the *Conversion Rate*, developers usually must give it context. For example, if 800 people click on an ad for a game, and get to the install screen, but only 40 of them end up buying the game, then the *CVR* for that ad is 5%.

4.5.3 CPI – Cost Per Install

Cost per Install is a measure of the total amount of money spent to market a game divided by the number of total installs. For example, if you bought 20 million impressions at \$1 each, and two million installed the game, then your total advertising cost would be \$20 million, and your *CPI* would be \$10. This measure is similar to *UAC*, but differs in that it typically is restricted to analysis of just the online advertising component of a project.

4.6 General Terms and Measures for Metrics and Analytics

Game developers use a variety of jargon when discussing and planning how to implement analytics and metric gathering into their products. In this section, I discuss some of the terms associated with telemetry (I am using the words telemetry and metrics interchangeably, please refer to Chaps. 1 and 2 for more details on their definitions) and analyzing user behaviors, which are commonly in use (see also Chap. 12 for more information on mining and analyzing telemetry data).

4.6.1 Auditability

Proper evaluation of data requires verification for accuracy against other highly trusted sources. The ability of a piece of data to be cross-checked is referred to as it is *auditability* (Meaning, “Can we audit this information?”).

Many third party analytics providers, such as Google Analytics, and the built in metrics provided by social networks, can provide good places to cross check data for auditability. Alternatively, cleverly embedded checksums in the analytics gathering code itself can be used to similar effect.

4.6.2 EED – Entry Event Distribution

EED deals with the first action users perform when they launch a game or site. By measuring and understanding this kind of user-behavior information, developers can determine what parts of a site or game are most effective. When players have a collection of choices what do they choose to interact with first? This can tell designers a lot about what keeps users coming back, or about order of play.

As an example, if 80% of users always first check the leaderboards to see how their ranking has changed relative to their friends, designers can assume that social competition is a huge motivator to their players. In contrast, if the users' first action is to check to see what gifts friends have sent them, they are probably playing for the cooperative social thrill. Every game will have different events and different *EED* measurements. However, this measure can be very helpful as it allows designers to learn to think in terms of common use cases and player motivations.

4.6.3 XED – Exit Event Distribution

What's makes a user quit playing? What is the last thing they do before they stop for the night? The answer to these questions could be very important, if they help the developer determine how to improve the product so as to increase user retention rates or the length of an average engagement. Studying the last thing a user does will not necessarily tell a designer what about the game is harming retention. The game flow itself may just encourage certain actions to come after other actions. However, studying the last thing a user does before they leave, never to return, is particularly important. Understanding common exit events will help give a designer insight into how they might improve a game to keep users engaged longer or reduce friction points. For example, if it appears to be very common that users go into a particular shop, but never end up buying anything, then quit the game never to return, a clever designer might seek to streamline the flow of the shop UI, or avoid forcing the user into that shop at that point.

4.6.4 Outbound Messages Per User (For Social Games)

Social games are really just those that allow and encourage users to make use of the communication features offered by the platform to help spread a message about the game. While initially, these were mostly on Facebook, or a few other

social networks, almost all platforms are (slowly) gaining a sense for the value of virality and community rewards. It is quite reasonable to expect that most mobile, platform, and PC games of the near future will begin to incentivize users to send out messages about the game to their friends in an effort to convert more users. Studying the number and types of outbound messages per user can help designers tweak their designs to be more infectious.

4.6.5 Message Conversion Rates

Most live games send out update messages to registered users through email or social media. The content of these messages can vary greatly, and it is ideal to be able to study how effective they are and tailor future outbound messages from the marketing department accordingly. Put simply, the more users who click on a link in a message, the more effective the *Message Conversion Rate* can be said to be.

Outbound messages generally come in two forms. These messages can be automatically generated by game events (or instigated by the user) and post a news item to that user's wall or social profile, e.g., "Jimmy just found a lonely sheep in Farmville." Alternately, the game can send a message directly to its users when they've not logged in for a while, reminding them to come back, e.g., "Play Farmville today and collect 20 cash and an English Hen!"

4.6.6 Virality

Virality describes the ways in which a game attracts new users: the rate at which the users who learn of the product become converts. Every element of the design and presentation of a game affect it is *virality*. The term virality is often used to specifically refer to the ways in which games that are played on social networks use the social network mechanics to attract new users. For example, early games on Facebook often issued many messages to users' friends in an attempt to entice new users to try the game. (And it worked so well, and annoyed so many Facebook users that Facebook ended up clamping down on many viral channels.) *Virality* is often referred to as the "K Factor," borrowing from epidemiology studies involving the spread of infectious diseases.

4.6.7 Engagement

Simply put, "user engagement" refers to how invested in playing a game the users are. The most measurable and objective determination of this tends to be how much time they spend playing. Moreover, since most free-to-play games don't make

money when their users aren't playing, driving up user engagement as measured in minutes per day tends to be important to most developers.

Studies of user engagement are a way of measuring the amount of time a user spends, per session, interacting with a product. The longer the player remains engaged, the more likely they are to spend money. Specific design mechanics greatly affect these values, and the types of targets that developers should set. Most games measure *engagement* in minutes and seconds, but since users can sit on some pages for hours without interacting, it may be more useful to measure the average number of page views per user, the total number of user inputs (clicks, keystrokes, controller inputs, etc.) in a given session.

4.6.8 Currencies

Many games use an in-game economic model by which a few different types of rewards are given out, e.g., gold, copper, experience points, rubies, etc. Games based on a micro-transaction model often have a couple of distinct types of currency that they allow players to use. Dual-currency models are particularly common, in which players are given in-game rewards in one type of low value currency, but can purchase higher value currency of a different type in exchange for real world money. For example, in *League of Legends* (Riot Games 2009), users win Influence Points by playing and winning games, but can purchase Riot Points with real dollars.

Understanding the relationship between different in-game currencies, and how they are awarded and consumed is critical to understanding how users interact with a game, as well as how they are monetized. As a result, many games build a significant number of metrics into their game to study the users' interactions with the game currencies.

4.6.8.1 Engagement Currency

When a game gives away a reward in-game, such as gold for slaying monsters, or IP in *League of Legends* (Riot Games, 2009) as a way of encouraging the user to perform a certain type of action, designers refer to this reward as *Engagement Currency*. *Engagement currency* is also often called "Soft Currency."

4.6.8.2 Hard Currency

Many online games, especially free-to-play or those that make their money from micro-transactions, allow users to speed their progress through the game, acquire items, or unlock certain features by paying for them with real world currency translated into in-game tokens. This type of currency is called *Hard Currency*.

4.6.9 *Retention Rate*

How many players return to the game for subsequent plays after their first experience with the game? The types of *DAU/MAU* calculation described above can provide hard data on this number, but when designers discuss it, they often refer to it as a game's *Retention Rate*. *Retention rate* is a measure of how "sticky" the game is, and can best be improved by polishing core game mechanics or reward mechanisms.

4.6.10 *Whales*

One of the dirty little secrets of free-to-play micro-transaction based games is that most users never pay, but some users elect to pay quite a lot. One of the great features of a free-to-play business model is that there is typically no ceiling on what a dedicated user can choose to pay in a given month. Designers typically find that their users fit into a few different spending bands, meaning that not all customers are equal. The top tier of customers, who represent fewer than 10% of a game's players, will often spend twenty-to-one that of the average player. Game developers call these high-paying users *Whales*. For a game in which the average user pays \$3 per month, a whale might end up buying \$1,000 worth of virtual goods. "Whale watching" is an art that many designers spend time on; after all, no one wants to make a change that will dissuade their most valuable customers from playing.

4.7 Why These Metrics Matter

Once upon a time, designers envisioned a game, coded it up, spent some time fixing bugs, and then released it into the wild to fly or fall in the marketplace on its merits. Things are different now. The widespread adoption of broadband in much of the game playing world, the rise of the online console, and the stellar examples set by a few metrics driven games companies (we are looking at you, Zynga) have changed the playfield. It is no longer enough to release a game – it must also be supported and modified after release in order to sustain a user community and reach the pinnacles of commercial success. The notion of "software as a service" (SaaS) instead of a packaged good has saturated the games market along with other sectors of the software business. To give customers what they want, we must study how they interact with the games and devices. Studying metrics allows game developers to detect problems and fine tune features and content. This highly reactive, iterative process is particularly important, especially for free-to-play games. For games, which rely upon tiny amounts of money from a very large number of users every few minutes, maintaining user engagement is of paramount importance. Modern game designers are wise to massage the details of their game mechanics and content over time as their knowledge of their user base grows.

When considering metrics, here are a few things to keep in mind:

- Tracking and gathering metrics is a relatively straightforward process, interpreting data correctly is much more difficult (as is alluded to in e.g. Chap. 5 of this book). This has resulted in the rise of experts in Business Intelligence, who (should) have superb minds for analytics and understanding statistical data.
- Developers should strive to structure their game metric gathering hooks to allow for easy experimentation, so they can test new theories, and gather metrics on new features and content easily.
- The most useful metrics tend to focus on making it easy to allow developers to test two different scenarios (A/B testing) and determine the effect each has on a particular metric. This leads to actionable information, which is easy to understand and verify. For example, when planning an email reminder to customers to try to bring them back for a new play session by enticing them with a free gift, developers might try sending out two versions of the mail to 10% of their user base. In one version, they may offer users two free in-game items, while in the other developers may offer users one in-game item and let them pick a friend to gift it to. Based on the number of users who choose option A over option B developers (or their marketing department) can determine which mail to send out to the remaining 90% of their customers. This sort of exercise is referred to as an “A/B Test.”
- The best analytics systems allow developers to gather and analyze metrics quickly. The goal should be the ability to pose a theory or hypothesis, instrument the game to test it, run a test, gather data, and make a change in hours or days, not weeks (see also Chap. 14 for an application in the context of AAA games; and Chaps. 6 and 7 for more on telemetry systems design).

4.8 A Case Study in Game Analytics Logic

In the following example, we will take a look at a hypothetical game, to analyze how we might use embedded analytic data gathering to help determine why the game is not more successful. This type of case study relies less on the use of the common metrics discussed in our definitions above, and focuses instead on the real-world application of metric analysis. While reviewing, consider how the types of gross metrics, which could apply to any product, are being customized to address the specific needs of this game.

For this example the game is a mobile tablet game designed for Apple iOS devices. It is a turn based board game, similar to Zynga’s *Words With Friends* (Zynga, 2009), but uses playing cards to simulate a variant of the game *Blackjack*. The game is initially free to play. Users can pay to convert to a premium version that is free from ads and offers other features. The lessons applied herein could apply to any turn based game, and could be modified to almost any game.

Rather than focus on specific implementation details, which are the province of the software engineer writing the code, we will instead look at the problem as a

diagnostic flowchart that a team could step through in order to evaluate potential points of friction that might be driving away users.

Problem: *Blackjack Buddies* is not making the expected amount of money. How might a developer study current user behavior in order to find out how to better maximize user engagement and spend?

1. Are enough people downloading the application? Review the total number of downloads from Apple, and compare this to the total number of people who launched the game for the first time.
 - If there is insufficient number of downloads, then this is a PR and marketing problem revolving around awareness.
 - Figure out how to improve the creative materials attached to the game, work with partners like Apple to promote the title, or figure out how to “re-skin” the game by giving it new artwork and a slightly different theme and re-launch the title with a sexier marketing angle, such as “Cards with Friends.”
 - Take out a small advertisement targeted for mobile phone browsers or clients from Yahoo, Google, or a similar search engine. Perform A/B testing on two different versions of the ad using just a few hundred dollars. Test CTR’s with ads that emphasize different features of the game, or potentially even with different names and taglines.
 - If initial review determines that plenty of users are downloading and launching the game for the first time then discovery is not the problem; time to examine other issues (see step 2).
2. Of those users who launched the game, what is the first thing they did in the game? (The Entry Event, remember *EED* Sect. 4.6.2) What percentage play multiple games through to completion? Note: add in the telemetry collection instrumentation the option of sending a message to the server every time a user completes a game.
 - If this is a majority, but most players are not choosing to upgrade to a premium version, you may have built a fun game that is capturing people’s attention but not monetizing well. In this case, you need to look at the value of the upgrades and figure out how to make it worth more to users or consider how the greater value of the upgrade could be more clearly communicated to users.
 - If very few users end up playing a full game, or most end up quitting after their first game, consider step 3.
3. In what part of the process are users quitting?
 - To determine this, add instrumentation each step of the way that reports the action a user just took. Because you can only really determine an Exit Event (see Sect. 4.6.3) based on the actions taken just before that event, you need to add a data point for each action the user can take. The more granular the better.
 - Then, since you cannot know for certain that a user will never return, you must filter the data for only those users who have not logged in again in a reasonable amount of time. Consider evaluating the average duration between user logins for those users who are active. Then multiply it by 10 to determine

that a user has dropped out “forever.” So, for example, if most users log in once per day, but a user has not come back in 10 days, it is a pretty safe bet you have lost them.

- How many users complete the game’s tutorial? If most users quit before finishing the tutorial, the tutorial itself might be adding friction. If the majority of users never complete the tutorial, you should consider making the tutorial an optional selection from the game’s main menu, and letting users jump straight into a game with friends or a bot.
 - If most of your users are getting past the tutorial, however, there are other issues to consider (see step 4).
4. When users try to start a game, are they succeeding in finding someone to play with? Evaluate your matchmaking process.
- What matchmaking option do users go for first? Battle a Buddy? Random Matchmaking? Solo Play vs. a Bot? Hotseat play with another local human?
 - Imagine that most users initially select to be match-made with someone they know using the “Battle a Buddy” button. When they do, what option do they select next to help them find a friend? These days, likely they look first to Facebook. When they select Facebook, and they are next led to a page in which they are asked to grant the application assorted permissions what percentage of players choose to Allow the app to do whatever it wants to do? If the number of DON’T ALLOW is a high percentage, then it is likely that the permissions page is frightening people away. In particular, common phrases like “Access my data any time” and “Post on my behalf” or “Bother all my friends” options are off-putting. If this turns out to be the case, consider making the application less intrusive and “spammy” so that it need not scare off your users.
 - How many players are making it through this process, finding a friend, then starting a game? You will be able to tell by reviewing how many people enter the Facebook section and comparing it to the number of players who actually begin and play through a game immediately afterwards. If this ratio is low then you have identified a problem in matchmaking and this is a good place to begin cleaning up the UI flow of the application.
5. When users get a game going (their first after tutorial), do they play to the finish? How many turns do they play?
- If they quit after a turn or two, then:
 - If they are playing vs. a human, AND the time between their first turn and their opponent’s turns is too long, it could be because the duration between the turns is off-putting. You should easily be able to check for this. What is the average time between turns?
 - If the duration seems reasonable but players do not end up playing more than a hand or two, it is likely a problem in the core game mechanics. This then requires you to delve deeply into the core game loop and ask hard questions about the gameplay itself, by setting up instrumentation or bringing in users for in-person playtests.

6. For those users who successfully get a game going and play through several games, then stop returning, consider the following.
 - Are these players winning or losing a majority of their games? If they are winning every game (either against bots or other players) or if they are losing the vast majority of their games, it is likely that they are quitting out of frustration, either because the game is too easy or too difficult. If they are getting matched against random opponents, you should be able to account for this with more sophisticated matchmaking heuristics. If they are playing against friends, there is less you can do. Either way, this is an area ripe for study with instrumentation and metric gathering.
 - And so on.

If this were your game, you would continue this process of coming up with a theory, instrumenting your software such that it would report back on user actions which would allow you to test the theory, then making a change in the software, and running another test until you had achieved the desired results. There is no end to this process, because a game can always be improved.

4.9 Conclusions and Next Steps

At its heart, the purpose of game analytics and instrumentation is, as described above, to allow the game's developers to gather sufficient data from real world user interactions with their product to enable them to make the game better, using an objective process of evaluation and iterative improvement. The various types of metrics I have discussed above are simply examples of commonly used types of data that can help developers understand their games better and make more informed decisions. While very few games are forced to use game telemetry or metrics, almost every game can be improved by studying user patterns and iterating on the software in order to give users more of what they want.

When developers look to evaluate a game, they typically start with a hypothesis, which is often formed as a question. They then determine the type of information that can best be used to prove or disprove their hypothesis. The case study discussed in this chapter serves as a brief sketch for how this process can work to help determine why a hypothetical game may not have as many users as its developers would like.

4.10 Additional Resources

There are several sources for additional information on the topics contained in this chapter:

- Sites like Kontagent (www.kontagent.com), Playtomic (www.playtomic.com), Game Analytics (www.gameanalytics.com), Playnomics (www.playnomics.com)

and Tableau offer middleware solutions that can help developers integrate metric telemetry into their social and mobile games. Moreover, studying the types of metrics that Kontagent and the other web/game telemetry providers offers can be informative. About two dozen such providers exist at the time of writing.

- Many game companies, like Electronic Arts' DICE studios, who build the popular online shooter, Battlefield, have issued reports on their game metrics. Check url: <http://publications.dice.se/publications.asp> or Ubisoft's Engine Room: <http://engineroom.ubi.com/getting-the-numbers-lessons-in-client-side-telemetry-gathering/>
- There are a number of blogs and online sites which focus on game data and telemetry. One of these is Blog. Gameanalytics.com which offers an excellent overview of the process here: <http://blog.gameanalytics.com/blog/2012/1/23/game-data-mining-fundamentals.html> and of course, the Game Developer Magazine online publication, Gamasutra www.gamasutra.com regularly offers insightful articles on these topics.
- For additional information on how to use telemetry and analytics in social, mobile, and free-to-play games, readers are encouraged to investigate this author's book, *Social Game Design and Monetization*, available from Focal Press.

About the Author

Timothy Victor Fields is a veteran producer, game designer, team leader, and business developer, who has been building games professionally since 1994. He has been involved in one way or another with several of the top franchises of the last few decades, such as *Need for Speed*, *Halo*, *Call of Duty*, *Brute Force*, *SSX* and others. In addition to making games and helping companies and teams find partners, Tim is active in the game development and financial community as a consultant, writer, and speaker. He has published two books in game development and production: *Social Game Design and Monetization* (Focal Press, 2011), and *Distributed Game Development* (Focal Press, 2009). He is a member of the IGDA, a SIGGRAPH speaker, and regular guest speaker at Texas A&M University and the University of Texas.

Chapter 5

Interview with Jim Baer and Daniel McCaffrey from Zynga

Magy Seif El-Nasr and Alessandro Canossa

Every day Zynga, the social gaming company behind hits such as Zynga Poker, FarmVille and Words with Friends, runs extensive statistical analysis of reams of data. Zynga has one of the largest data warehouses in the world, and the company needs it; its games generate over 15 terabytes of new data daily.¹ – says Praveen Asthana, *Forbes*

Zynga stores 1.4 Petabytes of data at any time and has the ability to deploy up to 1,000 servers in a 24 hour period. The company says the power they deployed for zCloud alone during the second half of 2011 could've kept 166 international space stations in orbit. 36 billion gifts were gifted during the holiday season in 2011.² – says Leena Rao, *TechCrunch*

Zynga has been on the forefront of using and establishing game analytics techniques both as a method to enhance the user experience and the design of games and as a business model to increase profit and make more successful games. Cadir Lee, Zynga's CTO, Jim Baer and Daniel McCaffrey have been instrumental at starting

¹ <http://www.forbes.com/sites/dell/2011/10/31/big-data-and-little-data/>

² <http://techcrunch.com/2012/02/15/zynga-ramps-up-private-cloud-infrastructure-zcloud-now-stores-1-4-petabytes-of-data/>

M. Seif El-Nasr, Ph.D. (✉)
PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

A. Canossa, Ph.D.
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

and leading the effort of this integration within Zynga. In this chapter, both Jim and Dan share with us a picture of how analytics is used at Zynga, the lessons they learned, and where they are heading.

Daniel McCaffrey, Senior Director of Platform and Analytics Engineering, has been at Zynga for a little over 3.5 years. He currently leads the platform and analytics engineering team. This includes analytics and the related reporting infrastructure in addition to platform services to support games, such as social communications.

Daniel started his career in science, specifically biotech. He spent 8 years in the biotech industry doing laboratory research. This is where he honed his analytics skills. He then spent several years in computer science in the field of bio-informatics. His work constituted developing computer science solutions to problems involving large data sets. He then co-founded a company called *BreadBoard BI* – a 10-person company – with the mission to provide affordable business intelligence (BI), data integration, and data warehouse consulting and training services.

Jim Baer, Senior Director of Analytics, has been at Zynga for a little over 3 years. He leads the use of analytics to improve the games and the business, supporting the different stakeholders who use the system. Jim has a PhD in Economics from Duke University. He was a professor in Economics at Emory University in Atlanta. In 1995, he decided to leave academia and join the analytics team as an Economic Analyst at the Federal Reserve Bank of Atlanta. Before joining Zynga in 2009, he had worked with multiple companies in the field of analytics. To highlight a few, in 2001–2007 he worked in a company called Telephia, Inc., which was later bought by Nielsen Mobile. During that time, he led the work on measurement science, specifically looking at the use of quantitative analysis of large data. During 1998–1999, he worked on target marketing at Abacus Direct. He was particularly using data to optimize targeting and driving. Before that, he was at BellSouth, again using analysis to improve the business.

Q: When you started at Zynga, was there an analytics group already or were you hired to start the effort?

Dan: When I joined Zynga in January 2009, there was no unified group working on business intelligence. Our CTO, Cadir Lee, was hired a month before me to build up the business intelligence group. Before that, there were some analytics, but the game studios operated somewhat independently. They would collect data on revenue metrics, but there was no integrated effort on analysis of large scale data, certainly nothing like what we have today. We started with very little and ended up transforming the company very quickly.

In the first 5 or 6 months, it was Cadir and I trying to prove the potential of analytics and its value to the various game studios at Zynga, not everyone was totally sold on it. However, there were some champions, certainly in the marketing team. We initially built the infrastructure starting with raw data, then made all the technology design choices. We took a different approach than what was currently used at the time, e.g., at Facebook and Yahoo or some of the other gaming companies. Our

approach was different in terms of the technologies and infrastructure choices we made, and it paid off in the end. Shortly thereafter we started building the analytics team to build the reporting infrastructure and analysis tools. This was when Jim joined the team in May.

Jim: Yes, I was hired just shortly after Dan started. In addition to sorting out the analytics and reporting systems, we built the analytics team as a central warehouse for all analytics tools and methods. As mentioned earlier, before that, all individual studios had their own solutions and worked independently. One of the first things we did was to develop this central warehouse and convince the various teams of the value of this unified solution.

Dan: Yes, this was actually the transformative part. Once we built the infrastructure, we then made it available; we opened the access to the raw data to everyone in the different studios. With the exception of the user demographic, id and revenue data, which we protect very tightly, data was made generally available. We were trying to let everyone get in there and experiment with the data, hypothesize and use the data to investigate user behaviors to improve their designs.

It is important to note that our analytics systems do not decide the future. It is the people, the designers, developers and other stakeholders who interpret the data and use the knowledge they gained from their experimentation to enhance the design. Analytics just provides a way to listen to the players, to see what works and what we may want to experiment on next. This approach was very successful in proving the value quickly to the different stakeholders.

Q: So, what kind of tools do you provide stakeholders in order to play around with the data?

Jim: what we have done is work with the partners in this 3-year evolution to understand their questions and what they want to see from the data, so that they can get at the question the way they want to. We have a huge amount of data and you can easily get lost in it. Thus, you have to be insightful on how you put it together to get at the information needed.

Thus, we developed some reports that our business and interested partners can use, based on the questions they wanted to answer. These reports were developed with the appropriate level of aggregation for the variables within the data. We also provided ways to allow stakeholders to build queries to access the data directly, because some people really wanted to do that. We showed them how they can build these queries to extract the kind of information they require. In addition, we built in support to do analysis and answer some of the ‘why’ and ‘what if’ questions.

Dan: I want to stress the point mentioned earlier. Instead of each group hiring their own analysts, we developed a unified analytics group who provides services to other groups in the company. This necessitated the need to make our systems usable and useful to different types of stakeholders. We also allowed access to raw data and ad hoc access to the database, to allow stakeholders to play around with the data. The analytics system has been so successful that it is now viewed as one of the three core competitor advantages for Zynga.

Q: Who are the stakeholders looking at the data?

Jim: Product managers, designers, producers, executives, developers, everyone, pretty much, but the artists.

Dan: Several additional teams, including finance, customer services, sales, business teams.

Jim: We are a feedback-driven company, and almost everyone wants to understand what data supports their suppositions, or what data they can utilize to answer questions.

Dan: And, how we can deliver services that players want. These are some of the main values that are driving our groups.

Q: Is there one tool that would fit all these stakeholders or did you end up developing different types of tools for different uses?

Dan: This is where we took a different approach than some of the companies out there. In our case, it worked out because the circumstances were right. We decided to use an ANSI SQL MPP database, Vertica, for the storage, instead of hadoop. That was a risk at the scale we knew we were headed to, but if we were successful, we knew the gains would be massive. The gains include ease of use, ease of integration with ETL and reporting tools, 10-100x faster analytics, less hardware, most analysts have the common SQL skillset. Three years later, and we claim it was a success for Zynga to take this approach and deal with data at a massive scale.

We integrated a system called Vertica.³ We worked with our vendor closely to redesign their system to suit our needs and deal with massive scale. The system supports full ANSI standard SQL and SQL analytic functions as well as regular text expressions, native SQL extensions, and a User Defined Function (UDF). These features worked out very nicely for all our needs and we were able to integrate it with our structure.

Q: Can you talk about the tools you developed?

Dan: We created a reporting tool, which satisfies about 90% of our reporting needs. But we also use Tableau and other tools to allow users to make up their own questions, and carve their own dimensions as they experiment with the data.

Jim: When we need more sophisticated visualizations, Tableau, is usually used to put data together to investigate relationships between the variables.

Dan: For more complex analysis like analysis cubes and visualization functionalities, we didn't want to build our own tools, when tools like Tableau exist and can provide that functionality. However, our in-house reporting tool has 2D charting and reporting, and satisfies almost 90% of our needs. We run about 5,000–10,000 reports daily. Our active daily users for our reporting tool, is about 300–500 users. ~700 different report types are run within a day. In a week, about half of the company is using the system, that is about 1250+ users. As you can see, analytics is deeply integrated within the company and used to assist in making daily decisions.

³<http://www.vertica.com/>, also see: <http://www.vertica.com/wp-content/uploads/2011/05/ZyngaSocialGraphing.pdf>

Q: Can you give us a broad overview of the subsystem for the analytics platform/service?

Dan: Most of the systems we have built from scratch with the exception of Vertica and Tableau, both of which we have licensed. We use several open source libraries, such as MySQL,⁴ MySQL cluster,⁵ Apache libraries, Membase,⁶ etc. We strongly believe in open source software. We also built our own private infrastructure called zCloud⁷ coupled with our use of the public cloud, our infrastructure is one of the world's largest hybrid clouds. However, we do still use Amazon Warehouse Services.

We have been giving talks to share these stories and what we learned.

Q: At the very beginning of the process of building the analytics group, what convinced the company to go that route? What value proposition did you give? Was it one thing or more?

Dan: For sure it was not one thing. What got people excited and lit the fire for the analytics integration within the company was: (a) access to raw data. This allowed people to log into the system a month later and look at the data in many different ways. Hence, product managers realized that they can access information about what users like, etc.; this was a big win. (b) Speed of access to data. We chose tools that allowed us to get to the data really quickly. From within 5 min of being logged, any team has access to it. This allowed us to do experiments and really quickly get back data to support or refute our hypotheses.

Jim: (c) The design of the system was very flexible. Thus, stakeholders can implement their analysis in the way they feel best suited their questions. The team can create their APIs to investigate the data they wanted. Since each game team can implement it the way they want it, it made it much easier to get them on board. (d) So simplicity is key here. We didn't give them a thick manual and tell them here is how you use the system. It was flexible enough for their needs and easy for them to integrate into our system. We would just give them guidelines and help them do it in a way that would be most valuable for them.

Dan: In most cases the flexibility and simplicity was a byproduct of the design choices we made developing the system. The system is semi-structured, we didn't build a fully structured database. Instead, we allowed users to create their own taxonomy that they then feed into our database, and thus our database is designed to be flexible and semi-structured in a way that it can accommodate the different users' taxonomies dynamically. We allowed users to log the data through APIs. We also provided reporting tools that allow them to slice through the taxonomies they built very quickly. This was a huge advantage, as it allowed them to quickly log new things and visualize the data.

⁴<http://www.mysql.com/>

⁵<http://www.mysql.com/products/cluster/>

⁶<http://www.couchbase.com/docs/membase-manual-1.7/index.html>

⁷<http://code.zynga.com/2012/02/the-evolution-of-zcloud/>

(e) Additionally, we allowed users to analyze on a massive scale, due to the sheer volume of data we collect. We have over 70 billion rows of data each day, and this data is available almost instantly. We have our own zcloud system. We also manage our own streaming solution allowing the data to be aggregated on the wire. From the moment it is logged, users can visualize it through reporting tools within a maximum of 1–2 min, versus 5 min in the Vertica based analytics system. This is mostly used for operationalizational and health reporting- presently. However, we have recently begun to use the data to do more real-time personalization and game adaptation dynamically at run-time. Thus, it is becoming an important aspect of our system.

Q: What type of personalization are you currently doing?

Dan: Match making is one major area. The nice thing about Zynga's games is that users do not have to share personal data to be connected or to play together with others.

Jim: Additionally, at a more general level, we can look closely at what people are enjoying and steer the content towards that.

Dan: Aggregation of content to personalized content in the game or in other places. But we also try to profile types of players based on their playing style and engagement. So we can do backend aggregation based on their playing tendencies so that we can match them more appropriately, but we can also score these results to use them at run-time to do different things, like targeting.

Q: What types of algorithms do you use for personalization and prediction?

Jim: Generally, we use many machine learning approaches looking for data relationships. We also use clustering for persona characterization.

Q: What are you working on now? Are you still working on infrastructure or infrastructure support? Or are you focusing more on personalization? Where are you moving as a team?

Dan: Analytics is the first service we had to create. We often have to build new services to leverage the data. For example, we have something that we call data services, as a generic container for data. Data goes into an analytics engine that scores different things, like retention, what the users liked, etc. We then push that into the network tier so that the game can access and make decisions based on that data at run-time. That is one service. Another is an experimentation platform that allows stakeholders to test and iterate on ideas very quickly.

Jim: The experimentation platform was very critical for us.

Dan: Yes, one of our philosophy here is testing and iteration. We need to be able to put features out within a restricted user group (e.g., internally within the company) and see how they are doing. Now that we have 3,000 people in the company, testing internally is a really useful way for us to test features. Also, when we roll out to the public, we also use experimentation platforms to roll out features and see how well they do. This allows us to iterate quickly with real feedback. Additionally, we also built cross promotion platforms to help cross promote our own games within the network.

These are all examples of systems we have been building. At this point there are about 15, of what I call, franchise services that are deeply integrated and shared within our company. Most of those leverage the analytics system.

Q: Is there any fear of overfitting or sampling issues with the data due to experimentation happening with a smaller group?

Jim: I don't know, is my short answer. We have a large number of people playing our games, so when we are doing any kind of modeling, we have a broad data set that makes it hard to run into issues like overfitting. There may be some problems or challenges with sampling, if we try to make some inference on a new game or a new experience. However, we don't take a sample of data from a game and try to predict what will happen if we put the game in front of a different set of people, or anything of that nature.

Dan: To add to what Jim just said, when we do internal testing, there is certainly a bias there. However, when we go externally, we have so many users that even a small experiment of say 1–5% of the total audience generally doesn't present any problems like overfitting. To your point, however, the analysis team here has worked out cook-books and training classes that we make available to people in the company on how to do proper experimentation. For example, don't just run your experiment 1 day in many cases, run it over a couple of weeks, so you can draw meaningful conclusions that account for weekly and time zone differences. There are guidelines that are shared within our company on how to run experiments and analyze data effectively.

Jim: It is interesting to hear the term overfitting. The paradigm has shifted so much these days. We used to do experimentation on such a limited sample that we worry about sampling and overfitting, but we now have an ocean of data with so many different types of users that these issues are no longer issues that are of concern when doing inference.

Q: What kind of advice about pitfalls would you give people who are starting out in this field?

Dan: I have been doing a lot of talks about this subject. I believe strongly, if your business allows for it, that having analytics as a centralized group is a big win. Most companies do the data collection in a central group and then have analysts spread all over the different groups of the company. In contrast, a centralized analytics group with people from that group embedded into the teams within the company is more advantageous. This is because it allows the company to share insights across the different groups, and also prevents replication of services.

Additionally, you need to have control over the data model with easy and flexible ways that people or teams can integrate their own taxonomies and query the data. This is very powerful, because it impacts the speed of how you can transfer and integrate the data across the entire network of products, instead of an open, log whatever you want, and then merge it all together; as in my experience, this strategy often breaks down with large problems.

In addition, the experimentation platform is important, but the point of giving people guidelines to make sure they do not fall into pitfalls with analysis is important. Jim has pushed this early on and has been instrumental in making sure his group provides training on how to actually do analysis in the right way, because most often people do not know how to do that, and they may be making the wrong decisions. This is not a really big issue, but we have seen cases, where people were making decisions too quickly on the data.

Q: Do your designers have any hypothesis on player models in terms of an ideal player behavior?

Dan: We do it differently. We do not have an ideal player model, instead we look for what the players like and then adjust for that. For example, instead of pushing for monetization, we focus on what the users value and make decisions based on that. We believe this will in the end impact monetization.

Jim: Yes, this is probably based on our internalization that there is no one user model. Rather, users are very diverse and interesting people, and so there is no one way to do things, different users like to approach the game in their own unique way. For a broad market game, instead of concentrating on a specific target market, we are listening to the players and building in flexibility so they can enjoy the game however they want.

Q: What are success stories of the use of analytics? What are the not so successful ones?

Dan: I think one success story we talked about is the idea of an active social network. Instead of looking at the total number of friends a player has in a game, you look at the friends whom participants are actually communicating with in games. This helps reduce unwanted communication. This was something that many people talked about as an issue. Facebook made some changes; we also made some changes. This was a big success.

The active social network measure turned out to be useful in other ways. It turned out that this measure was highly correlated with retention and other important variables for us. It was helpful at understanding if the users are getting what they want. The user value metric for us includes and leverages that.

Jim: An interesting discovery we learned was the fact that even though people may have several neighbors in a game, or other relationships within the game, there are far less number of people that they actually interact with, and those people (the ones they regularly interact with) are the most critical for the game experience.

Dan: For the not so successful stories, due to our experimentation platform we have many. There are many instances where we went one route and the data showed us that users didn't like that and we had to back track.

In terms of failure of analytics, one thing that we learned early on through several failures is to not focus on monetization or try to funnel users. Instead, turn this on its head and focus on how we can deliver something that the user wants or values and try to optimize and make more of that.

Jim: This is an important point. I want to go back to the point about advice and say that these types of analytics systems, like the one we have built, are incredibly powerful for tweaking the system and tuning it to get the best performance out of it. However, you can go down a rabbit hole doing that, because real innovation and real big changes do not come from those kinds of tweaks. Although they are also great experimentation tools, I would caution anyone who wants to do analytics for gaming to make sure they don't go too much into tweaking things and making those

marginal changes to try to grow their business, and remember that they need to come up with new ideas and test them out as well. This is one reason why Zynga has done so well; it is because we don't focus too much on tweaking things only but we also look for new ideas and test them out as well.

Q: What do you think is currently missing and where do you think the future is going for analytics?

Dan: Personalization and targeting. There is so much that needs to be done in this area. To your point, personalizing the games experience to what the particular user wants is one of our grand goals. The first key point for us to do was creating the analytics system and coming up with the infrastructure and the algorithms for that. As an industry, we are still scraping the surface in this area and we are always doing more and more on this front.

Jim: there is also some opportunity to broaden the data that we bring in to the analytics system. We started to do this, but there is much room to grow. We currently use some survey instruments to get feedback. The comments that we get from people are very insightful. Thus, we need some type of textual analytics to quantify and classify these feedbacks, integrate and triangulate that with the current metrics we already have. It is an incredibly exciting frontier to be able to integrate what people are telling us with what they are actually doing.

Dan: The qualitative and quantitative integration. We have done some of that with some of the games; we get qualitative data and combine it with the quantitative data. We find that to be very insightful and effective.

Q: One last question, you mentioned that you do some longitudinal analysis across multiple games, what kind of intelligence is useful to pass across games?

Jim: It could be many things, depends on what you are looking at. For example, you could pass information about action choices across games that are similar. A simple example could be, if you know that someone has spent time to decorate their board, then you may pass this information to other games that have that functionality.

About the Authors

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for*

Computer Games (Game Development Series) and Real-time Cinematography for Games. In addition, she has received several best paper awards for her work. Magy worked collaboratively with Electronic Arts, Bardel Entertainment, and Pixel Ante.

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and the Royal Danish Academy of Fine Arts, Schools of Architecture, Design and Conservation. His doctoral research was carried out in collaboration with IO Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches. His work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he maintains an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio.

Part II

Telemetry Collection and Tools

This part of the book focuses on a set of strategies, methods and tools with the purpose of collecting game data. Considering how different games can be, it might be unrealistic that a single set of best practices can exhaust the topic; therefore, key players in the industry have been asked to share their thoughts on data collection. Sony, Bioware, Unity Technologies, THQ as well as academic researchers have documented their telemetry systems' implementation and outlined development practices of their studios, delving into the lessons they learnt through the process, discussing successes and failures through a wealth of case studies.

This part has the following take-aways:

- Deployment of a telemetry system, from the data sources to analyses, integrated in an existing pipeline.
- Collecting and analyzing game metrics during the game production process.
- Strategies for monitoring the developers' use of tools for evaluating and enhancing the production pipeline, quality assurance methods, and workflow.
- Visions for inclusion of telemetry systems in a popular 3D engine middleware.
- Strategies for selecting the a representative and valid sample of users to collect data from.
- Design philosophy and concrete implementation of an open source API for logging, storing and analyzing telemetry data.

The part will consist of six chapters:

- Chapter 6: *Telemetry and Analytics Best Practices and Lessons Learned* is a contribution from Sony Computer Entertainment America (SCEA) by Sree Santhosh, senior Manger of Online Technology group and has contributed to many game titles at SCEA including Killzone 3, DanceStar Party, Resistance: Fall of Man, and Mark Vaden, Manager of Online Technology group who has contributed to many game titles at SCEA including KillZone 2, Syphon Filter: Dark Mirror. In this chapter they will discuss the telemetry system they developed at SCEA and lessons learned through the development of this system.
- Chapter 7: *Game Development Telemetry in Production* is a contribution from Georg Zoeller, Principal Designer and analytics specialist at BioWare who worked on several titles at bioware including Dragon Age, Jade Empire. In this

chapter he discusses the developer-facing telemetry and analytics system he developed and its use in game production. He will outline the process and the utility of the system.

- Chapter 8: *Interview with Nicholas Francis and Thomas Hagen from Unity Technologies* is an interview with Nicholas Francis, Chief Creative Officer of Unity Technologies, and Thomas Hagen, independent contractor working on analytics with Unity. The interview discusses Unity's plans to develop tools for game analytics and how they view the tools' utility.
- Chapter 9: *Sampling for Game User Research* this chapter outlines the benefits of sampling and the different sampling techniques and their use in game analytics. This chapter is a contribution from Anders Drachen and Magy Seif El-Nasr (editors of this book) in collaboration with Andre Gagné a user researchers at THQ.
- Chapter 10: *WebTics: A web based telemetry and metrics system for Small and Medium Games* is a contribution by Simon McCallum and Jayson Mackie both are at Gjøvik University College in Norway, and both have experience working on telemetry, databases and development of tools. In this chapter they will outline an open source middleware tool they developed for telemetry gathering and game analytics.
- Chapter 11: *Interview with Darius Kazemi*. Darius has over 10 years of experience analyzing game telemetry and metrics from casual games to AAA titles. This chapter outlines an interview with him to get his views on game analytics, the current state of the industry and where he sees the future of the field will go.

Chapter 6

Telemetry and Analytics Best Practices and Lessons Learned

Sreelata Santhosh and Mark Vaden

Take Away Points:

1. Keep the data flow simple and integrate early.
2. Decide which questions to answer before one starts collecting data.
3. Simple metrics while not as sexy as machine learning often provide immediate value that is easy to deliver.
4. Reliability of the gathering system is critical.
5. Cleansing of data is very time consuming.

6.1 Introduction

In this chapter, the process of embedding metrics collection within Sony Computer Entertainment will be outlined and discussed. We (the authors), are engineering development managers in an internal group within Sony Computer Entertainment that provides networking technology for games that Sony publishes. Our group provides a networking SDK and several online gaming services that benefit game studios in all Sony Computer Entertainment territories. We help integrate online services with games running on the PlayStation consoles, social networking and franchise websites.

The chapters in Part I introduced the basics of game telemetry and metrics. In this chapter, we go into more depth with the process embedding metrics collection in game development, specifically within the context of Sony Computer

S. Santhosh (✉) • M. Vaden
Online Technology Group, Sony Computer Entertainment America,
San Diego, CA, USA
e-mail: sreelatasanthosh@yahoo.com; Sree_Santhosh@playstation.sony.com;
markvaden@gmail.com; mark_vaden@playstation.sony.com

Entertainment. Similarly to the definitions in Part I, we define telemetry as the science and technology of automatic measurement and transmission of data from remote sources to receiving servers for recording and analysis. At Sony Computer Entertainment, we developed a telemetry system. This system was initiated in 2008. The type of titles we service affects our goals, and over the previous 3 years our technology has evolved, but our goal has remained constant. From the outset we focused on providing actionable answers to our customers' questions in a timely manner.

We are writing this chapter to provide some insight into what is involved with creating a telemetry and analytics pipeline at a company like Sony. Our chapter outline is listed here:

1. Definitions of a Telemetry/Analytics System
2. Design Considerations of a Telemetry System
3. Design Considerations of an Analytics System
4. An Overview of Our System Architecture
5. What Went Well With Our Architecture
6. What We Could Have Done Better
7. Final Advice and Next Steps

We feel we have built a functional system, but understand that there are always many design choices taken and revised through the years. We are in no way suggesting that ours is the best way, but instead are simply presenting our solution. We hope that this chapter will help you, the reader, in designing/implementing your own solution. After reading this chapter, you should have an understanding of what is involved in creating a telemetry and analytics solution. We believe we did some things well, and that by learning from our successes, as well as our failures, you should be in a better position than when we first started. In this chapter we will use our actual implementation as the conduit to instill this message. It is our hope that a real world example, with all its real world blemishes will provide a good example to discuss the pros and cons of a working telemetry and analysis system.

We begin by explaining our goals, and discussing who customers are. We define some of the terms we use, and what follows is a section that talks about design considerations for both telemetry and analysis. We then discuss our architecture, and finally close with a discussion of what is good about our system, and what we would like to improve.

6.2 Goals

The traditional method for data collection in the gaming industry, until recently, has been observance of a player playing the game in a usability lab, studying results from surveys or some combination of the two. These data points alone may paint a distorted picture, unless the hypothesis can be verified by the study of data gathered from gameplay. As discussed in previous chapters, analyzing gameplay data can help us prove or disprove a hypothesis and rule out common misconceptions of a

game designer, who may have a skewed view of the nature of his or her game. Oftentimes a visualization is so clear, that it dismisses any arguments. For example, heatmaps can very clearly show which parts of the map are well traversed, and we have found that these often contradict the designers' intuition.

As previously stated by Fields in Chap. 4, the goal of any data mining, telemetry and analytics service must be one of enabling the business to generate revenue by providing actionable metrics and consumer insights. In the competitive world that we live in today, there are many established video game companies that launch games of different genres year after year. Some factors that could make a game more successful are: appealing to a wider audience, well thought out game play mechanisms, higher quality, and an immersive community engagement. Therefore, in order to retain a competitive edge in the gaming industry today, it is important to understand how an end user experiences your game through the use of all available tools.

6.3 Potential Customers

As briefly discussed in Chap. 3 of this book, there are many potential customers or stakeholders of a telemetry and analytics service. We will share our experience in this discussion. In our experience, we have worked with many potential customers for our telemetry product; these include the game teams, game designers, senior management, publishers, marketing personnel, game producers, community managers and quality assurance teams.

For the game teams and game designers, the capture of player actions, movement data, and time spent on menus or game screens, player session information and a player's interaction with the rest of the community are some of the metrics that are worthwhile capturing and storing for later analysis. The analysis of this data can provide the game designers with a feedback mechanism to allow them to address unintended game play and improve the overall user experience.

The analysis of data captured by telemetry can help pinpoint choke points in a game, where player attrition occurs or where the player progression gets harder. When changes are made in a later iteration at that point in game, the game data collected may reveal advancement beyond that choke point. After validating that a choke point has been addressed, the analyst may continue to identify more choke points rather than continuing to address the first choke point after it has been fixed. This is an iterative process that continues until all choke points are identified and fixed. In social and online multiplayer games especially, player churn may occur rapidly in a short period of time and it is critical to be able to identify the issues quickly, and patch the game.

Potentially meaningful results that marketing personnel could derive from metrics are measurements that reveal characteristics of different player segments and populations. Different player segments may prefer different controller types, tend to consume different downloadable content associated with a game and experience micro transactions differently. The other areas marketing folks may also be interested in

could include an analysis of the conversion rates of beta or demo users to full paid purchasers.

Some of these studies may also appeal to publishing producers. If the game is being launched globally, publishing producers know that certain content and game play appeal to certain geographic regions in a globally launched game. He/she may be interested in introducing game content targeted to a specific region to make the game appeal to a wider global install base. Publishing producers are also looking to improve the game play experience, understanding how the end user experiences the game and preparing a roadmap of content to extend the shelf life of the game.

Quality assurance and usability teams often use the data to ensure coverage of test cases. One can capitalize on their test efforts for a title by pushing data from their tests run in a QA environment to game teams for an early evaluation of game balance, discovery of bugs and more.

6.4 Definitions in Telemetry/Analytics System

Game telemetry systems generally are client server based, since most of the time you are interested in data accumulated across multiple players. We use the term client loosely, and we will define it below. It is important to envision the system as follows: A game is played from a client. The telemetry data, as defined in previous chapters, is collected from the game (client).

6.4.1 Client

A client pushes information to a Telemetry server or set of servers. This information is usually accumulated from multiple clients. For example, data from multiple clients can be grouped together along session or game boundaries. A client can be programmed on top of many different platforms including: game consoles, game PCs, mobile devices etc. Clients can maintain persistent connections and stream data, or communicate in a request/response manner. Communication between the client and the telemetry server is largely one direction, with the bulk of the data moving from client to server, and the server only pushing commands.

6.4.2 Gathering

Collection or gathering of data is accomplished with a server to which clients connect. Gathering servers can be architected in many different ways; we will discuss details of how the data is stored, if it is processed, etc., in further detail later.

6.4.3 Processing/ETL

Data is often processed before it is placed into storage. This processing can be as simple as copying or forking a portion of the data. More complex systems may implement an ETL (Extract, Transform, Load) process. In a nutshell, ETL refers to the process of extracting data from a source outside of a data warehouse or database, transforming it to fit the final storage, and finally loading it into the final storage. This is a gross simplification, and a real-life ETL system usually involves considerable complexity. We decided to favor quicker turn around, and simpler implementation, as opposed to a more rigorous ETL system that could take years to implement; recognizing that we are risking some data integrity issues (e.g. data loss), we believe the windows to deliver game analytics are very narrow (for example, the stakeholder requiring an analysis may need a quick turnaround). Keeping the data processing pipeline simple, allowed us to perform data analysis work quickly and benefit from rapid iteration of defect fixing during narrow windows of opportunity, such as game beta phases.

6.4.4 Data Storage

Data storage can refer to both long term and short-term storage. We have found that in practice most game telemetry setups will probably use both types. Complexity and cost grow as the data grows. Therefore, it is sometimes beneficial to prune data where possible. Performing scheduled tasks that create frequent data rollups, remove duplicate entries, reduce data granularity over time are methods to keep data management easy.

The format of the data storage varies. It can be as simple as flat files stored on a file server, or database, to a data warehouse.

6.4.5 Analysis

Data is not valuable until it is analyzed. Analysis usually involves combining or transforming data into a chart, graph, table, heatmap or some sort of game specific visualization, which is then used to provide actionable issues to the stakeholders.

6.5 Design Considerations

When designing a telemetry system, there are many things to consider including: planned usage, cost, integration time, and the skill set of your organization amongst others.

Usage: First, you should define usage. While it is possible to create a system that handles every type of game, there are always compromises to be made. We have worked with games that send sub second event data, and conversely games that send user selections once every several minutes. These examples are intended to illustrate the differences between different game types. In an ideal world, you will have an idea of the questions you want to have answered before you begin your final implementation (see Chap. 14 for more discussion on this).

Cost: Another important consideration is cost. Different solutions have different types of costs. We found that cost is one of our most important design considerations. For example, consider assisting the game development team (as mentioned above) who were sending large amounts of event data. We realized we could save cost by storing the data in loose files. The advantage is that loose files are inexpensive; however, extracting the data from such files to answer a particular question requires parsing terabytes of data. Although such parsing issues can be circumvented through the use of a Map Reduce framework (Hadoop) to achieve reasonable time frames. Map Reduce has a higher engineering cost than running a query on a database. Such solution would work for a few fixed queries, and very few additional adhoc queries. Since this was the case for us, this was an acceptable compromise. However, in a situation where the game team is interested in joining lots of different data points, and looking at the data from many different angles, this probably would have been too expensive from an engineering standpoint. For example, performing a join in map reduce is complicated, therefore expensive in terms of engineering cost. Therefore, when attempting to consider cost, there are multiple solutions and compromises one can make, but you need to know more about the nature of use to determine what is the best solution.

Integration time: Integration time is also extremely important. At some point in every game cycle, tasks are cut in order to make the schedule. If you plan to post/analyze data from a game client, your window to integrate may be very small. It is important that your solution allows for easy integration. Our group provides middle-ware, and our first attempt at a client API was intended to give the customer a lot of flexibility. With this flexibility came some complications. We learned the hard way that teams under tight deadlines prefer simplicity to flexibility.

Organizational skill set: The skill set of your organization is another important factor. Internally, we constantly temper our enthusiasm for new technologies versus using technologies that we already have considerable expertise in. Of course, if we continue to use technologies we are comfortable with, our expertise in emerging technologies suffers. Thus, we try and use the right tool for the job, and if the timeline is short we will prefer a solution with which we are comfortable.

Data storage: Another consideration is how centralized you keep your data. We decided to centralize where our data is stored. We believe centralizing the data reduces the amount of time spent on ETL. With multiple data sources, and complicated tools, ETL/processing/parsing projects can consume massive amounts of time. We strive to deliver timely information, since our customers demand quick information. It is better to start small and answer fewer questions than to spend too

long and miss the period when the data would be useful. For example, studying game sessions, player engagement in game, player progression bottlenecks are examples of valuable analysis that can be performed quickly.

6.6 Analysis Considerations

6.6.1 *Augmenting Real Data with Survey Data*

While data mining and number crunching can tell us a lot about how the player experiences our game, it cannot tell us how each player feels at the end of the game experience or how to measure the player's emotional response to beautiful art assets, compelling storyline or exciting game play. This is critical as will be discussed by game user researchers in Part V of the book. Therefore, tying real game data collected during beta with survey data is invaluable.

It is important to decide if you are interested in either short-term game mechanic fixes or long-view profiling and modeling of data for a deeper understanding. If a long-term view is desired, such as studying various factors that may contribute to player churn, it means committing to collecting basic profile data prior to the start of the alpha or beta phase as well as soliciting input from players by presenting pop-ups with survey questions for more in-depth understanding. If longer-term items are not on the table, you may still want to run some beta test measures to filter and weigh by different player types.

Firstly, decide what things you want to know. Note that this is not the same as deciding what actual questions to put in surveys. Figuring out what intelligence we want first, and then working with someone to determine the best way to get the answers, is the way to go. In our experiences, working in parallel with a social scientist or sociologist who understands gamers and their social behaviors makes it easier to identify the survey questions that would help us glean meaningful information down the road.

In order to entice players to respond favorably to taking surveys, in-game incentives offered were often effective. If you can afford to have the luxury of migrating the incentives amassed by survey respondents to the game in production, it is often the most attractive incentive.

Some common things to remember when presenting surveys:

- If you use pop-ups, try to put them in the context of the game and in places where they are not a burden to the user, e.g. during a load screen when the player is waiting. It is important that surveys do not take away from the game play experience itself.
- Keep the questionnaires/surveys short and presented at the end of a checkpoint, game level or end of game (also discussed above).
- Keep answer selection simple. It should be dichotomous (yes/no) or multi-select. Use radio buttons, sliders, voice or simple text as input types.

- Keep data collection simple and focus on collection of a few metrics. Too much data gathered makes it harder to maintain the system, slower to run ad-hoc queries and may require dedicated time from a data analyst.
- Most importantly, make sure that the surveys can be connected and mapped meaningfully to actual data captured via Telemetry on the server side.

6.6.2 Dedicated Analytics Staff

Analyzing collected data can often be a daunting task for game teams who may not have dedicated staff or the necessary skill set. Demystifying the data requires knowledge of data analytics and investment in full-time analytics experts to realize the full benefit from the data collected.

In order to hire the necessary talent, it is often required to justify the initial investment in these resources by presenting a projected economic impact to justify increased spend. However, our experience has shown that game teams, that have dedicated personnel looking into game data and sharing findings with the designers to incorporate into their next game revision, help bring in incremental value over time to cover the extra costs.

6.6.3 Using Sampling or Altering What Data to Collect Dynamically

The most common challenge with telemetry is the fact that data collection can occur rapidly, and often all at once especially at launch times or just after a game update. Soon, rather than spending time with data analysis, the analytics team is busy spending a large amount of time in housekeeping; preparing scripts to trim data growth, computing roll-ups of data and running cleanup scripts on files. In order to keep costs down and for the purposes of practicality, it is often good to limit data collection to a sample of the entire player population. There are many methods used in sampling, as covered in Chap. 9 in this part of the book, like random sampling, cluster sampling (that separates the population into meaningful groups) or stratified random sampling (that separates the population into mutually exclusive sets). For more details on the different types of sampling techniques please refer to Chap. 9. However, random sampling works well enough for most practical purposes. It is important to determine the sampling error early on and adjust sample size accordingly.

Another method of limiting data collection is coding in the ability to configure data collection in your telemetry solution. This allows one to dynamically turn on and off certain metrics from being collected. This is useful since the metric needs may vary during the life span of the title.

6.6.4 Types of Reports Requested by Customers

The moment a game begins to collect data, customers of this game data are eagerly seeking new ways to leverage the large and ever-growing stores of high-quality game data that are available. This power to harness data should be put to use immediately. We have observed that while there is a considerable amount of overlap in the metrics (we also call them data points) that may appeal to the different consumers of the data, there are also distinct differences in the data types that interest the different groups of consumers.

Some common metrics that are collected by most games irrespective of genre are game player session information, player progression data, network statistics, player metrics, player social interactions and commerce transactions. Notably, some of the most common report requests that came our way were:

- Trend data by week or month for the lifetime of the game that shows the usage of features/games/game assets
- Player session information
- Player progression choke points
- Daily or monthly active users
- New account creations weekly, monthly
- Reports on ‘conversions’ from demo purchase to a full paid version of the game
- Reports of content downloads broken down by region
- Ability to run ad hoc queries
- Simple reporting mechanisms that can act on a subset of data

6.6.5 Differing Data Types Appeal to Different Game Data Customers

While there is often significant overlap in metrics captured and consumed by different groups, we have found that even within a game team the different groups of users may consume different data points for analysis.

In our experiences we have found that **game designers** love to understand game metrics such as player session information, game maps and modes played, as well as specific player metrics such as level and rank progression. Other metrics that were interesting to game designers were:

- The characters encountered
- Boss fights
- Objectives earned
- Weapons used
- Assets consumed by career type or as a player levels up

A game designer then uses the information to adjust the game play mechanics to make it more appealing and keep the player optimally challenged and therefore playing longer.

An **online engineer**, on the other hand, may be more concerned about data capture relevant to network analytics. Metrics of interest to a network programmer are:

- Frame rates
- Upstream bandwidth
- Memory usage
- Screen load times
- Queue wait times
- Measurement of effectiveness of the game's matchmaking algorithm

A **producer** is usually keen on measuring the effectiveness of the commerce elements in game. Therefore the category of data that would appeal to a game producer would be:

- Study of a player's purchase history
- Transaction details
- Items browsed in the store or added to cart
- Player behavior by region
- Length of player engagement with the title
- Consumption of downloadable content
- Overview of additional incremental review

Unlike all these stakeholders, a **community manager**, who is the voice of the company externally, as well as the voice of the gaming community internally, has many needs. He/she has a strong presence on forums, blogs and news articles. Therefore, he would often request data that appeals to the wider gaming community, such as lifetime global stats of the game, like:

- Total number of games played lifetime
- Total player hours of game play
- Maximum points earned by the community overall
- Top players
- Observed glitches in game and positional information
- Shortest time taken for an objective earned
- How many players have finished the game or reached veteran status

6.6.6 Type of Game Determines When the Data Should Be Delivered

The gaming industry is in a state of flux. There is a proliferation of smart phones and cheaper (even free) gaming options available to the end-consumer today. Social gaming via Facebook and other social networking sites have taken away some market share from traditional console based gaming. Even within the console based

gaming industry, there is a subtle push to extend the lifetime of a game with the introduction of downloadable content and expansion packs. It is the hope that in the future, this will allow for incremental monetization several months after revenue from initial sales is realized.

One key observation that can be made is that the revenue source for the game largely determines the timing for the data delivery and analysis. If the primary business model that drives sales of a particular game is from revenue realized by the initial retail sales of the boxed product or initial download of the game via a digital download store and less from ongoing content purchase, then the reports required must be identified upfront. The end-to-end process of data capture, storage, analysis and reporting must be working well in advance of the title launch. In this scenario, the data gathered during QA, usability tests and beta testing phases are invaluable to the game.

On the other hand, as introduced in Part I of this book, for social games many of the metrics are time based and related to measures of acquisitions, retention and virality (average revenue per user (ARPU), rate of signups, % conversions). Some social games try and segment populations to different server sets to provide them with different gaming experience. This form of testing is called A/B or multi-variate testing. The data collected from these two or more varied content gaming experiences are studied to determine which one produced increased conversions. Many social games do a social network analysis over time and measure how the game changes over time by studying:

- The mapping of player interactions
- Predictive analysis of human interaction to identify subpopulations
- Key game mechanics such as stranger interaction
- Player kick and ban events
- Events like ignoring other players
- Voice communications

The data therefore needs to be provided constantly over a period of time, with game mechanics tweaked and results recalibrated to derive optimal results.

One of the common challenges faced with data collection in games is that it is often one of the last technologies integrated into a game after code completion of the game is complete. Also, the engineer integrating the data gathering technology into the game code base is often working on his/her own, and often not all of the reporting requirements are understood upfront. In order to get the maximum use of the data once it is captured, it is necessary to design the reports alongside the capture of the metrics, identify all metrics that are required early on and draw parallels between survey data and actual player data for later comparisons.

6.7 Our Architecture

Our customer base consists of console game developers, so our architecture is tailored to console games. It could be tailored to fit any purpose, but some design decisions would certainly change. Our architecture consists of a client library,

a gathering/collection server, a customizable visualization tool, an Oracle database, a high-speed file server, inexpensive file storage, and a Map Reduce cluster.

6.7.1 Client Library

Our client library allows developers to quickly integrate metrics into their game code. Our architecture is built with a concept of streaming data as opposed to sending discreet events. This works well for our customer base, since most games usually have a time span that defines a session. In an online game, this normally coincides with a particular game session. This works well for our customer base, since most games have a start time and an end time that defines a game session, and data from discrete game sessions can be isolated and streamed into individual files. For single player games, we usually define a session as the time a player plays a particular level. Our client is data agnostic, so this concept of a session is arbitrary and ultimately left up to the person who is implementing the metrics or the stakeholder of the data. Our client library is also cross-platform compatible as it runs on game consoles, Linux, and Windows.

For games that have a server maintaining a game state, it makes more sense to have the server post data, because the game server and gathering servers should be able to exist on the same LAN. For games that have a server maintaining a game state, it makes more sense to have the server post data to the Telemetry gathering servers. Bandwidth costs can be reduced when both game servers and Telemetry gathering servers exist on the same LAN.

The client sends data in a binary format. Game console based clients are likely to be bandwidth sensitive. Rather than having all of our customers implement their own routines to pack metrics into as few bits as possible, we decided to add this functionality to our library. The end user describes how many bits he/she wants to use for a particular value and type, and we pack it. For example, for packing an unsigned integer, the end user could specify that he/she wants to only use 4 bits, with the obvious consequence of having an integer limited in range to 0–15.

Our client also supports sampling. The server controls the sampling rate. We put the server in control. This allows us to change the sampling rate after the game is in production without having to patch a game client, since most of the code that has to change is in the server side.

6.7.2 Data Gathering Architecture

We had two requirements for the gathering portion of our system. First, it had to scale in a cost effective way. Second, it needed to be very reliable.

Our gathering/collection server was implemented in C++. We chose C++ mostly due to us being most comfortable with the language, but we did think that it was possible that we could make use of the better performance of C++ over managed

languages. However, we suspect that our primary bottleneck might be I/O, in which case language choice is arbitrary. We kept this server deliberately simple. It handles 4 tasks: authentication of the client, managing connections, decrypting data, and writing data to disk.

Client authentication is probably the most complicated portion of our gathering server. A server certificate authenticates the server to clients, and a client certificate authenticates a client to servers. Authenticating a client or server using the digital certificate involves three steps. First, the party with the certificate sends it over the wire in the open, to the other party. Second, the received certificate is parsed and the signature verified for authentication. Third, the first party proves that it owns the associated private key. Once the authentication is complete, the gathering server establishes a connection.

Establishing a connection is relatively straightforward. Clients request a connection on a known socket, and after authentication occurs a socket is created for future communication. All communication with this client happens on this TCP socket. We use TCP, rather than UDP, because we buffer enough on the client to live with the header overhead, and we make use of the built-in error correction feature of TCP.

Once a client has established a connection, it can start sending data. In our system the stream of data has a beginning and an end, and we call distinct streams “frames”. A client initiates the start and end of a frame. In between the start and the end is the client data. Data is buffered locally on the client, until the payload is large enough, or the client ends the frame, and the client sends the data.

When a client requests a frame to be created, the server creates a file on an NFS mount. Every piece of data is written to disk without the server looking at the data. This allows the gathering server to be data agnostic. It just takes whatever data it receives and writes it to disk. This simplicity created a reliable mechanism for gathering data.

When we optimized this server, we wanted to take advantage of multi-core machines. To do this we split our server into three types of threads. The first thread type manages the connection request/authentication. We wanted the authentication in its own thread since the authentication is one of the few CPU intensive tasks that exist on the gathering server. The second thread type is responsible for managing the data that comes from each socket. After prototyping what we guessed would be some typical traffic, we concluded at any given time many of our connections would be idle. We decided to serve many clients’ sockets with each thread, and use non-blocking I/O and readiness change notification. The mechanism we used for this is *epoll* (Kegel 2006).

Our final thread type is used to manage the disk I/O, and we spent a lot of time optimizing our disk usage. The major lesson we have learned is that you need to be flexible in your threading model to take advantage of different hardware. Sometimes we are fortunate to be connected to a clustered NAS system. In this case we want to try and run as many things in parallel as possible, so we split the writes into multiple threads, as well as separate the opens. When writing to local disk this approach was slower, since our data was written sequentially. We separated opens into a different pool of threads, but left the writes in a single thread. The takeaway here is that you need a clear picture on the hardware you plan to ship on, and adjust your implementation accordingly.

Our gathering servers sit behind a load balancer, and any gathering server can communicate with a client. Furthermore, if a client loses its connection to a particular gathering server, it can reconnect to any other gathering server. This flexibility improves load balancing.

6.7.3 *Processing Architecture*

Our processing requirements had three main requirements:

The first is that the data should be parsed once, and exported or forked to multiple consumers. Consumers could include databases, data warehouses, custom visualization tools, etc.

The second requirement was for the system to be data driven. We wanted to be able to handle almost any data sent to us without having to make code changes. We wanted our configuration to specify the layout of the data, the type, and the amount of data.

The third requirement was scalability. We wanted to be able to add more processing servers based on need. For example if we had two servers processing the data, and they were not meeting the time requirements, we could add a third. As much as possible we wanted our processing servers to scale linearly.

We again used C++ to implement our processing servers. Our processing servers understands the file format of the gathering server frames. Moreover, it understands the directory structure that the gathering server uses. The only communication between the processing server and the gathering server is through the file system.

The processing servers walk the file tree and find a file that needs to be processed. Every file written contains some information about the file, including a game title and version. The game title and version is used to look up an xml file that contains a description of the data being sent. By parsing an object type field, the processing server can look up the data format of a particular object type. An example of our data description xml follows:

```
<ObjectType name="GameStart" id="0" export="True">
  <DataMember name="ip_address" type="kType_IP"
export="CSV"/>
  <DataMember name="build_number" type="kType_Int32"
export="Oracle"/>
  <DataMember name="tsd_name" type="kType_String"
export="Oracle"/>
  <DataMember name="game_level" type="kType_String"
export="Oracle"/>
  <DataMember name="game_mode" type="kType_String"
export="Oracle"/>
</ObjectType>
```


The object starts with an object type, which allows the processing server to look up the object description. Following that is a list of fields, each with a type. This allows the assembly of an object from the binary data. Notice the export attribute in the xml. In this example, the IP address is going to be exported to a CSV file, while the other fields will be exported to an Oracle database.

The processing server has a configuration file that maps export types to a particular plug-in. When the processing server finds a type with an export attribute, it passes that data to a plug-in written to handle that particular type of export. This gives us a lot of flexibility; we can easily add another plug-in to export to a different storage type, and by simply updating the configuration, and changing the xml, the processing server will invoke that plug-in.

Like our gathering solution, our processing server is also multi-threaded. There is one main thread that handles the crawling of the file directories, as well as the parsing of the file. The main thread also looks up the appropriate description xml. Once the data has been placed in memory in the correct format, the main thread passes the data into N number of export threads that are each running a specific plug-in for the type of export required. Once the data is exported, we are ready to run reports and perform analysis.

6.7.4 Analysis and Warehousing Architecture

Primarily, we store our data in two different ways: in an Oracle database, or in the original binary format in which it was sent. Data that we plan to join in many different ways, or that we need to query frequently, we store in the database. This ends up being a small subset of the total data. The remaining data we leave in the binary format and archive onto slower, cheaper storage. For example, key metrics that provide metadata information is inserted into the database whereas positional information is kept in raw format in files residing in cheaper storage to be consumed by other data visualization tools. We will discuss the tools we use to analyze the data in Oracle in a later section.

The data stored in the binary format is analyzed in different ways. One way that game teams get good value from telemetry is by analyzing their ongoing play tests. While many solutions exist for analyzing standard data, game teams often require custom visualizations. After evaluating different off the shelf solutions, we decided to make a custom visualization framework that could be quickly modified to provide any visualization. Our visualization tool, a screenshot of which is shown in Fig. 6.1, is a Windows GUI application written in C# using .NET and WPF (Windows Presentation Foundation). Although it is able to display some visualizations right out-of-the-box, the real power comes from writing custom visualization plug-ins.

An example of a custom visualization is shown in Fig. 6.1.

This visualization takes movement data in a video game and provides a heatmap (see Chap. 16 for more on heatmaps). Across the bottom of the image is a legend that specifies the color values. The “hotter” areas coincide with more movement. The “cooler” more blue areas have less movement. While many data visualization

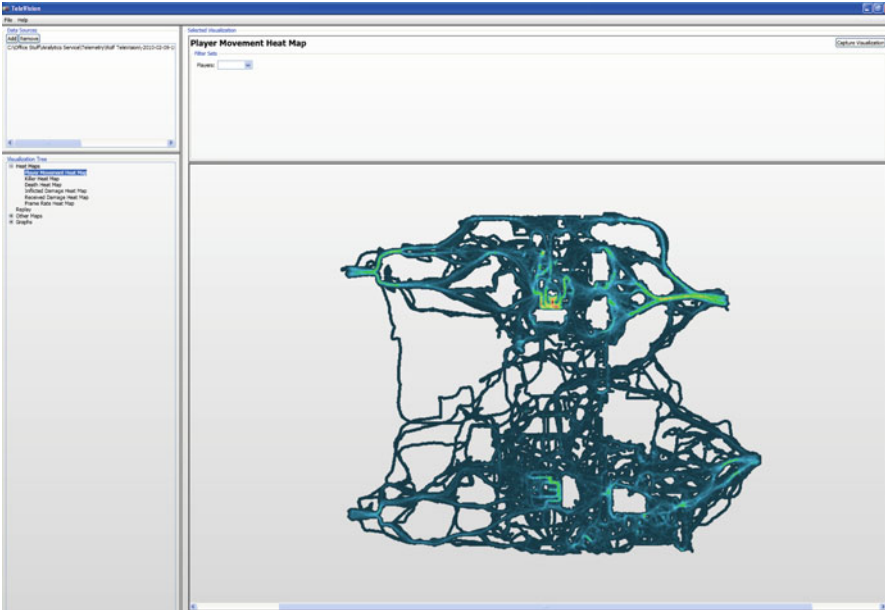


Fig. 6.1 TeleVision Movement Heat Map, showing the trajectory density of a game level (Image courtesy of Sony Online Entertainment, © Sony Online Entertainment)

tools now support heatmaps, we have found that by making a very simple framework that is catered to game teams, we can provide a solution that fits more with how a game studio operates.

Our custom visualization tool uses the same data description XML that our processing servers use, so there is no need to rewrite parsing code. Along the left side of the image, you can see a tree control that defines which visualization is active. This visualization tree is a hierarchical list of your visualizations. A visualization is any visual representation of your data – e.g., a line graph, bar graph, heat map, table, etc. The visualization tree is defined by an XML file, and is specific to your title or application. Example Visualization XML file:

```
<TeleVision>
  <ImportManager                dll="dataimporter.dll"
class="BinaryImportManager">
  <Parameter name="Schema" value="XMLData definition.
tsd"/>
</ImportManager>
<Visualization name="Heat Maps" type="None">
  <Visualization name="Movement Heat Map" type="Bitmap">
    <Importer                dll="                dataimporter.dll"
class="DataImporters.HeatMapImporter"/>
  </Visualization>
</Visualization>
```

```

    <Visualization name="Killer Heat Map"
type="Bitmap">
    <Importer dll="DataImporters.dll"
class="DataImporters.KillerHeatMap
Importer"/>
    </Visualization>
</Visualization>
<Visualization name="Graphs" type="None">
    <Visualization name="Kills by Weapon Type"
type="ZedGraph">
    <Importer dll="DataImporters.dll"
class="DataImporters.KillData
Importer"/>
    </Visualization>
</Visualization>
<Visualization name="Tables" type="None">
    <Visualization name="Weapon Stats" type="Table">
    <Importer dll="DataImporters.dll"
class="DataImporters.WeaponTable
Importer"/>
    </Visualization>
</Visualization>
</TeleVision>

```

The visualization XML contains the name of the dll to load during the visualization, and the class that performs the importing. There is also an optional visualization class, but in practice most of the time the same class is used. C# has reflection, which makes instantiating a class from XML very easy. This xml customization allows an individual game studio to have its own unique tool with minimal amounts of custom work.

Occasionally, a game team wants us to do some analysis across terabytes of files still in a binary format. An example of this would be trying to find information about how frequently a player is killed within a certain distance of a spawn point. To accomplish these types of tasks across large data sets, we use Hadoop Map Reduce. From the Hadoop website (Apache Software Foundation 2007): “The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.”

Hadoop is easy to get started with, and most of the really hard distributed problems are solved by this framework. We started with Hadoop with no expertise, and quickly were able to parse small amounts of data; however, when we needed to run our same Map Reduce jobs over large data sets, things often went awry. If you have never used Hadoop before, plan on a long ramp up time to debug your first complex job across a large data set.

One of the biggest challenges with data collection and analysis is the volume of data that the system needs to handle. While our telemetry pipeline stores all the raw data in files on storage, it also inserts a subset of key metadata into a relational database. In our case we have used Oracle® as our transactional OLTP database. The data is also replicated to a data warehouse environment nightly, in an automated manner, to allow further data modeling and separation from the transactional environment. Canned reports are built in our OLTP environment using SQL queries and exposed to the end user via a IBM Cognos® web portal for data visualization in the form of pie charts, histograms etc.

More complex adhoc queries and models are built in our OLAP data warehouse environment (also Oracle®). We have recently successfully experimented with the creation of a multidimensional, 64-bit, in memory OLAP cube using (IBM Cognos TM1® 2012) that provides reasonably fast performance on large game datasets. While one is limited to the available machine memory, one can perform reasonably fast analysis after streamlining the data load process to only pull in delta records nightly.

There are quite a few benefits to invest time in data modeling:

- The end user can perform independent analysis on the data.
- It caters to end users like producers who can use drag and drop mechanism to view result sets.
- Less time is invested in writing custom queries, testing and promoting new packages to the transactional environments.
- There is less risk to the dataflow pipeline as most of the data analysis occurs on a non-transactional environment.
- Studios that are keen on accessing their data have a low risk mechanism to view their data in a non-transactional, data warehouse environment.

6.7.5 *Testing*

Our testing was broken into two types of testing. We performed functional/unit testing as well as load testing. Functional load testing was the responsibility of the engineer who implemented a feature. Load testing was performed from the moment that we had the basic connection system running, through the end of the project. We had dedicated hardware for load testing, consisting of a set of machines acting as servers, and a set of client machines that run hundreds of clients.

We wrote our own software internally to manage load testing. Our load testing architecture is built in Linux. Our basic architecture consists of a single process that manages the whole load test, which we call internally the simulation manager. In addition to the simulation manager, each load test machine runs a daemon that is responsible for spawning instances of the client test process.

When a load test is started, the simulation manager pushes the test client binary to each client simulation machine. The daemon running on each client simulation machine will spawn the configured amount of instances of the client process.

In addition to pushing client binaries, the simulation manager will also push configuration. The configuration is used to control the client behavior. A common use of the configuration is to create a probability based state machine, giving our clients more flexibility. An example of this is our load test of client connections/disconnections. We had a client that read from the configuration, the probability of a client disconnecting before sending any data.

In addition to configuration, our load testing system also supports logging. Each client logs to a central log on the client machine. The logs are stored in a binary format, and the writing of the logs is highly optimized. The logs can be converted to text, and combined across multiple client machines.

6.7.6 Monitoring

6.7.6.1 Internal Monitoring

We realized that we would need a quick way to monitor our load tests. We built a system that allows us to quickly check the health of our servers during a load test. Each server is connected to a monitoring server. The monitoring server contains a round robin database, and stores various metrics. Metrics are broken into OS level metrics, as well as application level metrics. Examples of OS level metrics include CPU usage, memory usage, and socket level metrics. Examples of application level metrics include number of frames open, number of frames written, current number of connections, number of disconnects, etc.

Our monitoring system visualized the metrics stored in the round robin database via a web page that displays configurable line graphs with the x-axis being time, and the y-axis being the metric desired. A screen capture is show below. By quickly checking a web page we could get an overall view of the health of the server.

6.7.6.2 Production Monitoring

Our internal hosting group has a whole set of hosting related tools base around the tool Traverse (Zyrion 2010). Our internally developed monitoring tool will not work in our production. Therefore, we added functionality to pass server health related metrics in a protocol preferred by our hosting group (Wikipedia 2008).

6.8 What Went Well with Our Architecture

While we had some things we would do differently, which we will address later in this article, our first attempt at a Telemetry system was a success. We provided valuable information back to game teams that integrated our technology. Our first pro-

duction launch coincided with the launch of a Sony PS3 game, and we were extremely stable. Several things that we did right contributed to our success.

6.8.1 Separation of Gathering and Processing

As mentioned earlier, our gathering technology is separate from our processing. This ended up being a good design decision. One advantage is the simplicity of the gathering servers. Parsing data has a higher risk for bugs. In fact, we had some crash bugs in our processing servers, but since they are separated from the gathering servers, we fixed the bugs, redeployed, and the data was parsed and exported with no data loss.

Another advantage of the separation of processing and gathering was freeing up cycles for the gathering server. The connections and traffic to the gathering servers can sometimes be bursty. When we have a high amount of traffic, the gathering servers keep writing data, and the processing servers fall behind. If they start to fall too far behind, we can deploy more; frequently they catch up on their own once traffic lightened.

6.8.2 Load Testing

We started load testing early, and we load tested frequently. We found and fixed many bottlenecks. In fact, we continue to load test even now in spite of the maturity of our system. Running and maintaining a load test can be time consuming and tedious, but the up side is enormous.

6.8.3 Data Driven Development

Our export tools, our load testing, and much of our technology are data driven. The benefits of being data driven are well known but sticking to your data driven guns is often difficult. The temptation/pressure to break your data driven paradigm “just once” is hard to overcome in the face of timelines, pressure, etc. We are not suggesting that we never hard coded anything, but our team stayed true to the data driven ideology.

6.8.4 Custom Visualization Tool

Although many visualization tools already exist, we found that game teams were not very interested in using them. We believe that an extremely simple framework

that allows game teams to make extremely focused analytics is the best way to accomplish “play tests” analytics.

6.8.5 Leveraging Existing Technology

In what appears to be a contradiction, leveraging existing technology exists in both the “what we did well” section, as well as our “what we could improve” section. The reason it exists in the “what we did well” section, is because we’re very successful in leveraging technology that our group had created. Our load testing framework, our internal monitoring tool, and in fact the core of our servers were pieces of technology that our group had developed for previous projects over the last 10–15 years.

6.8.6 Separate Transactional Database and Data Warehouse

We started out writing custom queries in our transactional database and soon discovered that it was much more efficient to extend it to create a data warehouse solution which provides ad hoc query capabilities to the game teams.

6.9 What We Could Have Done Better, or Ongoing Work

6.9.1 Testing

We never integrated an automated functional/unit test system. Leaving the testing up to the individual engineer predictably resulted in less testing during crunches. Our group has technology for functional and regression testing, but we never integrated it. The result was that we found many bugs in load testing that could have been easily found in much less time with better unit testing. One of our current tasks is to integrate our automated functional test system.

6.9.2 Leveraging Existing Technology

There are several commercial and open source projects that could have helped with pieces of our pipeline. This could have saved time, and in some cases may have even resulted in improving the product offering. We did not research the options enough, due in large part to the do it yourself type culture that exists in the Video Game world.

We might have been able to do our data gathering using Amazon Simple Queue Service (Amazon SQS). Additionally, (Tableau 2012; QlikView 2012) are available commercial off the shelf analysis and visualization tools that could be used instead of our own visualization tool.

6.9.3 The Cloud

We are just now starting to experiment with using cloud solutions, like Amazon's EC2, instead of internally hosted machines. Other groups within our group have had good results with using EC2 for load testing. We made some technology decisions that make our servers less efficient in EC2. We are in the process of optimizing for cloud deployments. In hindsight we should have recognized the momentum cloud solutions were enjoying, and designed with cloud and custom hosting solutions in mind. We are paying the cost of this now, having to rewrite some core components of our technology.

6.9.4 Production Monitoring

We underestimated how long it would take to integrate metrics into the Traverse monitoring system that our hosting engineers prefer. As a result, some titles launched without any monitoring causing our development team to monitor servers. The monitoring was manual, and was work that none of us were excited to tackle. If we had integrated the monitoring systems earlier, this work could have been done by hosting engineers who are better qualified and already organized into 24-h support.

6.9.5 Administration Tools

There are several tasks needing to be done on a regular basis that we have not automated. We designed a system to provide automatic to near automatic administration of our servers, but we never implemented it. In hindsight, things were prioritized over these tasks that have turned out to be less important.

Some of the admin tasks that could have been automated with a tool include, migrating older data to cheaper storage, reprocessing raw data, query the system for parsing/processing errors.

6.10 Final Advice and Next Steps

If we were to start all over again, we would do some things the same, and other things differently. What follows here are the things that we think are most important.

First, integrate early. Get some metrics integrated into your game today. Getting two or three valuable metrics in early is easy, and allows you to start on your pipeline. Likewise, start your analysis early. As soon as you start getting data, start analyzing it. This will help you plan out your data flow. Additionally, things missing will become readily apparent.

Combine your data with some sort of survey information. You are really limiting what you can understand without some sort of qualitative data. Frequently in the games we work on, the qualitative data is not survey based but just oral feedback from a play session. The qualitative data gives the quantitative data a framing, allowing better decisions to be made.

Make your analysis/metric gathering actionable. We have found that what is interesting does not always correspond to what makes the games better or the company more money. What is actionable is often game specific. For example, console games get most the benefit from early testing, and beta testing. In the case of franchises there is also benefit for planning sequels. Other game genres, like MMOGs and social games, have opportunities to monetize over a wider time period, so the data gathered needs to reflect those opportunities.

If you are interested in pushing this type of technology further, we would recommend exploring non-traditional methodologies for data stores. We touched upon Map Reduce, and readers interested in quickly integrating this type of technology would do well to consider HIVE. More information can be found at: <https://cwiki.apache.org/confluence/display/Hive/Home%3bjsessionid=1646DEFA8C182D53C2CF1F30A6CB9EC8>. Also Amazon has added some useful features to Hadoop as part of their Elastic Map Reduce. We believe that the integration of their other services with Elastic Map Reduce makes for a powerful platform. For more information: <http://aws.amazon.com/elasticmapreduce/>.

About the Authors

Sreelata Santhosh is Sr. Manager of Online Technology at Sony Computer Entertainment America and a technology management professional. She has expertise in SaaS solutions, integration of network services into games, analytics, on-demand software, high volume transactional websites and services, software migrations/upgrades, product development and professional services management. Sree and team have worked extensively in online game development for PlayStation VITA, PlayStation 3 and PlayStation Portable. Her team has created and maintained many networked services and solutions and helped integrate them into SCE games. Online Games that Sree and team have contributed to, include:

Starhawk (PS3), Sony Computer Entertainment America, Inc.
Killzone 3 (PS3), Sony Computer Entertainment Europe Ltd.
DanceStar Party (PS3), Sony Computer Entertainment Europe Ltd.
Socom IV (PS3), Sony Computer Entertainment America, Inc.
MotorStorm (PS3), Sony Computer Entertainment Europe Ltd.
MAG (PS3), Sony Computer Entertainment America, Inc.
Resistance: Fall of Man (PS3), Sony Computer Entertainment America, Inc.
SingStar (PS3), Sony Computer Entertainment Europe Ltd.
Killzone 2 (PS3), Sony Computer Entertainment Europe Ltd.

Mark Vaden is a Manager in the Online Technology group at Sony Computer Entertainment America. His love of data was the motivation for co-founding the software company DashWare (www.DashWare.net). In addition to his work in telemetry and data, Mark has been programming video games professionally for nearly 20 years, specializing in telemetry, online technology, and graphics. He has contributed code for the following video game titles:

Starhawk (2012), Sony Computer Entertainment America, Inc.
Killzone 3 (2011), Sony Computer Entertainment America, Inc.
MAG (2010), Sony Computer Entertainment America, Inc.
MotorStorm (PS3), Sony Computer Entertainment Europe Ltd.
Killzone 2 (2009), Sony Computer Entertainment Europe Ltd.
Syphon Filter: Dark Mirror (2006), Sony Computer Entertainment America, Inc.
SOCOM 3: U.S. Navy SEALs (2005), Sony Computer Entertainment America, Inc.
SOCOM: U.S. Navy SEALs – Fireteam Bravo (2005), Sony Computer Entertainment America, Inc.
Twisted Metal: Head-On (2005), Sony Computer Entertainment America, Inc.
Ape Escape Academy (2004), Sony Computer Entertainment Incorporated
Backyard NBA Basketball (2003), Atari, Inc.,
NHL FaceOff 2003, (2003), Sony Computer Entertainment America, Inc.
NHL FaceOff 2001, (2001), Sony Computer Entertainment America, Inc.
Road Rash: Jailbreak (2000), Electronic Arts, Inc.,
Road Rash 3-D (1998), Electronic Arts, Inc.
Tiger Woods 99 PGA Tour Golf (1998), Electronic Arts, Inc.
Magic: The Gathering – Battlemage (1997), Acclaim Entertainment, Inc.

References

- Apache Software Foundation. (2007). Hadoop MapReduce. Hadoop. <http://hadoop.apache.org/mapreduce/>. Accessed Sept 2010.
- IBM Cognos® TM1. <http://www-142.ibm.com/software/products/us/en/cognostm1/>. Accessed Sept 2010.

- Kegel, D. (2006). The C10K problem. Dan Kegel's Web Hostel. <http://www.kegel.com/c10k.html>. Accessed Oct 2008; (2009) epoll. Linux Programmer's Manual. <http://www.kernel.org/doc/man-pages/online/pages/man4/epoll.4.html>. Accessed Sept 2009.
- Oracle Standard Edition One. (2012). <http://www.oracle.com/us/products/database/standard-edition-one-066500.html>
- QlikView. <http://www.qlikview.com/>. Accessed 21 Mar 2013.
- Tableau. (2012). <http://www.tableausoftware.com/>. Accessed 21 Mar 2013.
- Wikipedia. (2008). Simple Network Management Protocol. Wikipedia. http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol. Accessed Oct 2008.
- Zyrion. Traverse Quick Start. Zyrion. <http://www.zyrion.com/support/quickrefs/traverse-quick-start.php>. Accessed 21 Mar 2013.

Chapter 7

Game Development Telemetry in Production

Georg Zoeller

Take Away Points:

- Presentation of a Bioware’s *SkyNet* telemetry system, which tracks player behavior as well as developers’ usage for the purposes of evaluating and enhancing not only game design but also the production pipeline, quality assurance methods, and workflow.
- Discussion of the implementation of the *SkyNet* system as part of the game development process during the production of BioWare’s *Dragon Age: Origins* between 2007 and 2009.
- Summary of experiences and lessons learned through the process of deploying the SkyNet telemetry system

7.1 Introduction: The Development of a Developer-Facing Telemetry System at BioWare

There are different types of telemetry and metrics systems employed within the game industry, reflecting the varied usage scenarios across different game forms and formats, as well as the different stakeholders involved. In this chapter the main focus is on telemetry applied towards developers, here termed “developer-facing telemetry”, and the system we have established at Bioware to facilitate developer-facing telemetry. The main goal of the system is to facilitate and improve the

G. Zoeller (✉)
Ubisoft Singapore, Singapore
e-mail: georg@gulbsoft.org

production process, gathering and presenting information about how developers and testers interact with the unfinished game. This contrasts user-facing telemetry, which is typically collected after a game is launched and mainly aimed at tracking, analyzing and visualizing player behavior (see Chaps. 4, 5 and 6). In this chapter, I describe the developer-facing component of the SkyNet system developed at Bioware, and our experiences developing and using the system.

Understanding of user behavior on earlier BioWare Projects, such as *Jade Empire* (2005) and *Mass Effect* (2007) was mostly derived by analyzing end user achievements or by analyzing log files created by internal testers, providing only relatively basic information on the user's interactions with the game. For non-persistent game worlds on consoles, such as the *Mass Effect* series, telemetry systems are usually not as extensive as the telemetry collected for social online or persistent titles. This is due to the fact that it is not always cost-effective to modify a non-persistent console game after release. Hence, the telemetry data is less useful. Data collection and analysis often focuses on understanding customer demand for post-release content or lessons for the development of sequel products rather than the continued development of the game itself.

During the production of *Dragon Age: Origins* (2009), we decided to introduce a developer-facing telemetry gathering and analysis system that was intended for use during the production process. Therefore, we chose to instrument the game with a detailed metrics solution during the production process, gathering information on how developers and testers interacted with the unfinished game. These metrics helped us improve our production processes, and, ultimately, had a measurable impact on the stability and quality of the final game.

Our telemetry system, internally dubbed 'SkyNet', started as a simple, network based event logging system developed in-house. It was initially located in a simple desktop machine under my desk consisting of a simple collection server that stores triggered events, sent from the game, in a relational database. After the release of *Dragon Age: Origins* in 2009, we graduated the server into a datacenter. The simple small database became a larger database filled with more than 250 GB of gameplay data. Many of our internal workflows changed for the better based on what we learned from the data.

It is this experience and lessons learned from it that I aim to discuss in this chapter. I will start with a discussion outlining and defining developer-facing telemetry systems in terms of their requirements and goals. I will then outline the system we developed at BioWare. Subsequently, I will follow with a discussion of the lessons learned from the deployment of SkyNet, in particular its impact on the workflow and quality assurance pipeline.¹

¹ It should be noted that all tracking of telemetry and usage discussed in this chapter was disclosed and happened with the consent of the users/employees.

7.1.1 *What Is a Developer-Facing Telemetry System?*

While sharing several similarities, developer-facing telemetry and end-user facing telemetry are fundamentally different in the sources they collect data from, the types of telemetry that is tracked, and the application of the collected data. First, developer-facing telemetry systems are intended to gather data from a few hundred developers, as opposed to millions of users for end-user telemetry system. Telemetry is also collected for a limited amount time within the cycle of game production. With a constrained data storage capacity, this can be seen as an opportunity to collect as much data as possible from early users (developers and playtesters, in this case), without having to worry about scaling the system to millions of potential end users. The same level of depth (or resolution) may not be possible for end-user telemetry systems, this is due to many reasons: (a) the process of data collection can sometimes impact game performance which is an important issue, (b) the games are often persistent which impacts the number of data collected as it will grow through time presenting scalability issues. These issues constrain end-user telemetry system in terms of the level of detail and total amount of data that can be collected from individual users.

Second, end-user telemetry for real world players is subject legal issues on top of the storage and bandwidth considerations, such as regional privacy regulations. Such regulations ultimately limit the type and volume of collected data. Only some of these issues arise when collecting data during production.

Third, when analyzing data from developers, we have the liberty of resetting an entire event database (telemetry storage) several times when we discover better ways of approaching a particular problem. This is something that is rarely feasible for a post-release metrics system targeting real customers.

Fourth, with a developer-facing telemetry system it is often beneficial to introduce new telemetry hooks² on the fly whenever there is demand. Being able to create a game event hook and receive data from it within the hour was a huge benefit during performance optimization, as it allowed us to capture data for any performance relevant metric almost instantaneously. For a shipped non-persistent title, this is significantly harder and usually involves costly patches.

7.1.2 *SkyNet Infrastructure*

The architecture of the SkyNet system is shown in Fig. 7.1. At the core, it is a simple message event logging system with database backend. Various clients (game clients, tools) send network packets with event data to a central tracking server which

² A piece of code instrumented to trigger a message to the telemetry system when executed, resulting in a game variable to be monitored and measured, generating metrics.

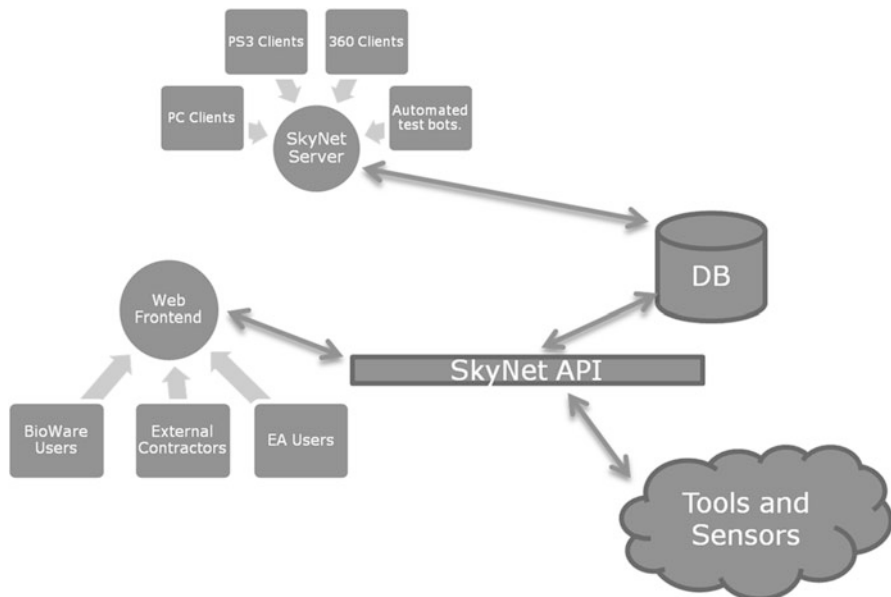


Fig. 7.1 BioWare 'Skyner' systems architecture (Used with permission from Bioware)

handles decoding and data submission. As shown in the figure, messages are transmitted from the game to the collection server (using UDP network packets). While UDP carries a risk of packet loss, we chose this protocol over TCP due to its high speed, ease of implementation, and because the majority of the data captured is stored within EA's reliable internal network, which made packet loss generally unlikely. We made the conscious choice early in the project to consider most of the data captured by the system as 'non-mission critical' and consequently did not implement measures to guarantee transmission to and storage on the server. In retrospect, even though we lucked out and did not experience any data loss, we should probably have opted for a more reliable form of transmission. Packets are dispatched from the game using a single, unified interface and are interpreted and stored by the collection server using a set of precompiled scripts. This creates a pluggable infrastructure where new packets can be added, removed and changed on the fly without restarting the server. For data storage, we used a standard relational database server (MySQL) with ample of disk space.

In order to track time spent within the game and provide context around the individual interaction events of a game session, each game is identified by a unique session ID generated at the beginning of game execution. This also provides valuable context for developers in case fatal events, such as game crashes, stack dumps or memory leaks, occur. In order to announce itself to the tracker, each game sends an initial session start packet to the server upon startup.

Although the kind of telemetry data collected within the developer-facing telemetry system is similar to the end-user telemetry system, there are some differences. Due to reasons discussed above, we had little constraints on what we can track. For the *Dragon Age* project, we instrumented more than 140 unique game events, allowing us to create a highly detailed picture of users' actions and behaviors while interacting with the game. This level of detailed logging is generally not feasible with end-user telemetry – not just because of privacy concerns, but also because of the pure volume of data generated by it, as discussed earlier.

For each session, data collected includes:

- Session ID – A unique identifier for each individual game start that provides context to any other event captured within the same play session.
- User ID – Uniquely identifies the player by their active directory username. This field is used in a variety of ways (personalized reports, attributing defect tickets, etc.).
- Platform – Identifies the SKU (PC, XBox360, PS3) of the session.
- Machine IP/Hostname.
- Build/Version Number, Language, etc.
- Start Time/End Time/Total Time/Time Spent Idle.

Once a session has been established on the tracking server, the server starts accepting game events from the game client. Game events in our system represent user actions or the results of user interactions with the game. These events are identified by a unique ID per type along with spatial coordinates. Information about the originating object and optional parameters are stored and time-stamped, in sequential order along with several optional parameters.

In addition to the session data discussed above, game event-based data is also collected. For *Dragon Age: Origins*, we tracked dozens of different user interactions with the game. Examples include:

- Player character death, enemies defeated, etc., for purposes of game difficulty balance.
- UI Interactions to understand how players interact with the game interface and when they rely on certain interface functions (such as use of the in game map system).
- Economic events, such as treasure or gold collected, which proved vital for balancing the in game economy.
- Player movement/stuck events, which allows for better understanding of how players travel through the game, where they might be able to unintentionally exit the boundaries of a game area, etc.
- Player customization choices (e.g. which weapons they utilize on their characters, which class or race their characters are, etc.) to ensure that internal testers cover all content in the game.

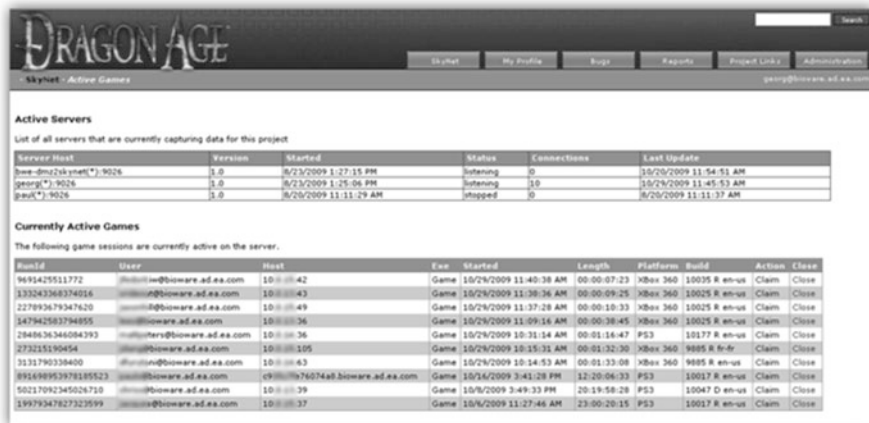


Fig. 7.2 The ‘active games’ dashboard for Dragon Age: Origins, showing presently played games across the studio (Used with permission from Bioware)

7.1.3 Visualization and Reporting Subsystems

One of the most important lessons we learned about metrics over the course of the project was: without a good way to visualize and explore the collected data, the data is almost worthless. Initially, most reporting was done by manually extracting information from the database and analyzing it using Excel or simple scripts. This was fine when most of the data collected was analyzed by the relatively limited number of people who requested it. However, as more stakeholders showed interest in using the system, this quickly became too labor intensive and impractical. In response to the increasing demand for data access, we deployed a ‘self serve’ web frontend, written in Asp.Net and running off a stock webserver, with dashboards (see Fig. 7.2) for different departments and ability to publish reusable reports to stakeholders.

The project that started with a handful of technical designers and engineers providing developer-facing telemetry and analysis soon became bigger, servicing multiple stakeholders from different departments. Managers in Quality Assurance requested information about the interactions of testers with the game. Localization producers expressed interest in understanding exactly which parts of the game were used by foreign language testers. Writers wanted to understand which dialog choices were the most popular among early playertesters, and so forth. Our primary concern while developing and iterating on this system was not so much usability, but rather how to reduce the amount of work required to create the diversity of reports requested by the stakeholders. Over time, this interface morphed into a complex visualization tool with support for filtered reports, graphs and spatial event visualization, such as heat- and point plots using in-game maps. Figure 7.2 shows an example simple visualization of time spent by players from another EA studio.

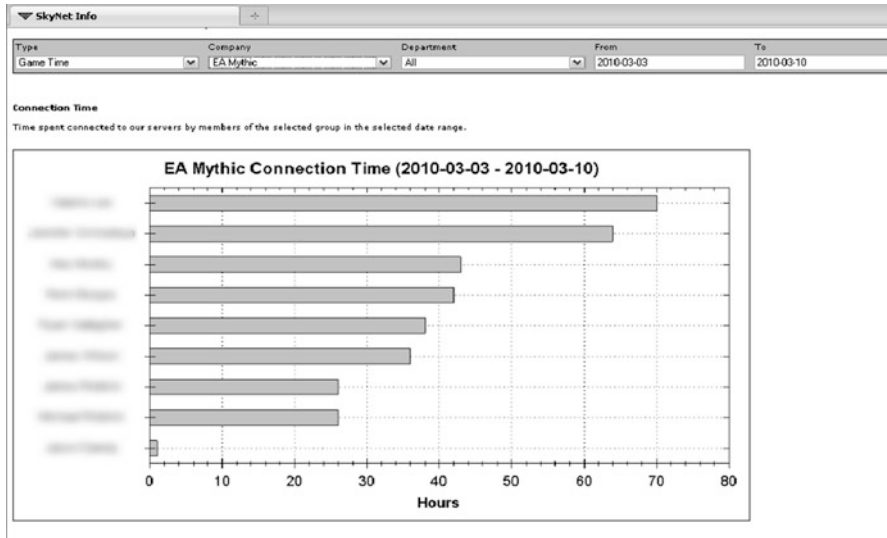


Fig. 7.3 Sample of a filterable report visualizing time spent with the game by players in another EA studio (Used with permission from Bioware)

Due to the increased number of stakeholders, we had to find ways to supply different types of reports that satisfy the different needs (example in Fig. 7.3). In particular, we found ‘drill down’ reports very useful to supply information on various levels of the project and organization (see also Chap. 12). For example, executives wanted high-level summaries, while developers wanted more detail. Using ‘drill down’ reports we can supply high-level information to executives, and such data can then be expanded to generate detailed reports for team leads and individual developers. A good example of such a report is the Build Integrity Dashboard seen Fig. 7.4, which provides a highly simplified top level overview of the technical status of various build SKUs. Drilling down into the graphs delivers progressively more detailed information to the developer (see Sect. 2.1).

While it was certainly enlightening to create an entire reporting frontend for the metrics database from scratch, we ended up spending a significant amount of time on maintaining and extending it. Today, powerful and customizable off-the-shelf solutions such as Tableau³ or the Katana Engine⁴ will likely provide a more long term and cost effective visualization solution for most developers.

³ <http://www.tableausoftware.com/>

⁴ <http://ninjametrics.com/>

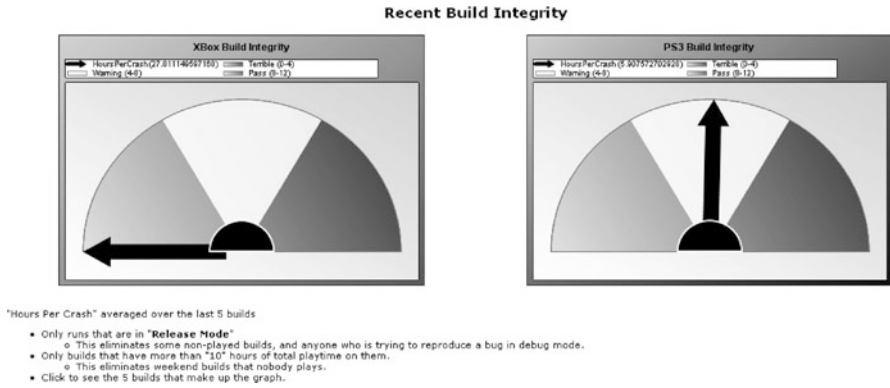


Fig. 7.4 The Build Integrity Dashboard (Used with permission from Bioware)

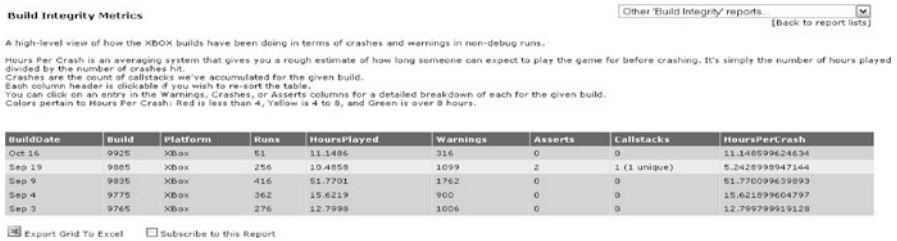


Fig. 7.5 Build Integrity Metrics dashboard (Used with permission from Bioware)

7.2 Collecting Metrics to Benefit Development Pipeline

7.2.1 Benchmarking

On previous projects, when trying to answer the question 'Are we ready to release?', we would use a combination of open issue reports, quality assurance reports and gut-feeling to determine the answer. Due to the existence of concrete data, for *Dragon Age: Origins*, 'gut-feeling' was replaced by a high level 'Build Integrity Metrics' dashboard using an aggregate of automated crash reports, warnings, asserts and tests coverage metrics for each area and platform (Fig. 7.4). The release readiness question could now be answered to a much higher level of confidence. On the topmost level, the Build Integrity Metrics gauge (Fig. 7.4) presents one simple, color coded gauge for each stock keeping unit (SKU), summarizing the current 'release readiness' of the project into a simple three state display. By clicking on each chart, the user can drill down into the more detailed metrics used to compute the chart. Figure 7.5 provides an example table displaying how long the game can be expected to run before crashing and other stability measures.

Several, progressively more detailed layers of drilldown are available, ultimately down to the level of each individual game started during the measurement period.

7.2.2 *Production Pipeline Metrics*

Game companies spend significant resources trying to make the production process more predictable; BioWare is no exception. With *Dragon Age: Origin*, we started to systematically gather metrics from our tools to better understand usage patterns, identify performance bottlenecks and detect workflow issues. With several hundred developers contributing to a title like *Dragon Age: Origins*, it is easy to lose track of proprietary development tools and scripts and who depends on them. Working on video games always means dealing with unstable and unfinished code, in both tools and game. Developers quickly get in the habit of working around issues in creative and time-consuming ways and, in the light of more pressing, unavoidable issues, it can take months or years to address these problems. While we had initially only gathered usage metrics from the game itself, we later realized that gathering usage metrics from our tools held significant potential for identifying these kind of hidden productivity killers.

By tracking interactions with the content development toolsets, we were able to measure the frequency and length of use for each tool and establish its importance to the overall production process. Later, we expanded the data tracked by adding detailed metrics on user interaction for individual functions of the toolset to identify time-intensive operations that were used with high frequency (such as generation of pathfinding data, etc.). In the following subsections, these will be discussed in more detail.

7.2.2.1 **Measuring Tools' Stability**

One lesson learned during the production of *Jade Empire* was that content developers rarely complain openly about the tools' stability. Faced with unstable tools, users would quickly develop work around for the stability issues, by frequently restarting the tools and invoking time consuming save functions more often. Ultimately, this behavior caused a silent loss of productivity until discovered later during the project.

On *Dragon Age: Origins*, we were able to uncover this problem by collecting and analyzing data on usage patterns and stability of the content developer toolset and, over time, enabled us to determine the best places to apply engineering efforts for maximum impact. See for example, Fig. 7.6.

7.2.2.2 **Measuring Productivity**

Once usage metrics for toolset and game had accumulated for a few weeks, it became possible for us to extract and aggregate usage patterns that provided further

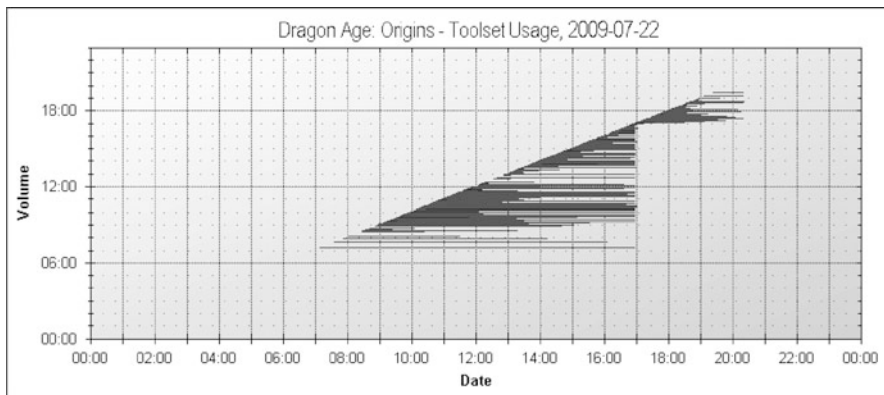


Fig. 7.6 Daily usage graph for the Dragon Age: Origins Toolset – Each *horizontal line* (x) marks an individual session of the toolset (start to end), while its position on the y axis also marks the time of day the tool was started. *Vertical cutoffs* indicate a server outage that affected a large number of users (Used with permission from Bioware)

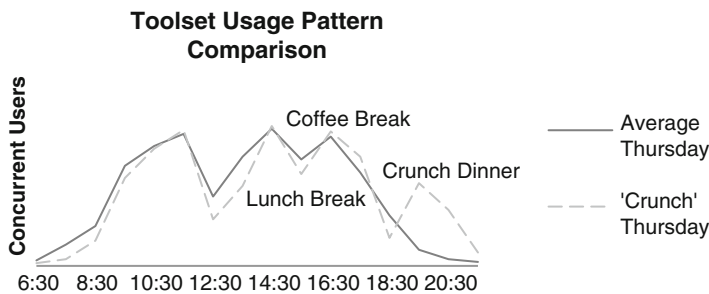


Fig. 7.7 Toolset usage pattern comparison during development, indicating the relative minimal difference in the use of SkyNet during normal days and crunch days (Used with permission from Bioware)

insight into our production process and the general efficiency of content developers. We can now put a cost (lost man-hours) onto events, such as power outages, office moves and team meetings, and understand their impact on the schedule. It also enabled us to measure the success of ‘meeting free Thursday’ – an initiative to reduce the fragmentation of the workday by declaring meeting free periods during the week. Finally, it put visibility on the diminishing returns of overtime periods (crunch time) from a behavioral perspective. Figure 7.7, for example, shows the difference in the usage pattern between a normal work day and a crunch. While certainly providing a measurable productivity boost in the short term, after a few weeks of crunch time, developers would inadvertently start compensating for lost downtime by taking extended lunch, coffee and dinner breaks – along with a general shift to later work hours.

7.2.3 *Quality Assurance*

Dragon Age: Origins was a massive project, even by BioWare's standards:

- 80+ hours of story with 800,000 words of voiced dialog contributed by 144 voice actors
- 300,000 lines of designer script code, 3,000 cinematics, 55,000 animations and roughly 20,000 art models.
- Fully localized release in three language on Xbox 360, Playstation 3 and Personal Computer.
- Build from scratch: New game engine technology, new game ruleset, new intellectual property.

At BioWare, everyone in the studio, from administrators to programmers to studio managers, is encouraged to play each game before launch. This enables everyone in the studio to provide feedback and report defects, and often provides another layer of quality assurance and polish for our games. However, the scope of the project posed a unique problem for our traditional quality assurance process. Even with a large number of experienced career QA-professionals on staff and access to Electronic Art's Canada based QA resources, there was no way we would be able to establish full path coverage for all the permutations of the game's story line and game areas for each milestone, let alone for individual builds.

In order to remedy this problem, we had to be creative. Using production telemetry, we implemented several measures to increase testing of *Dragon Age: Origins*. This included a dedicated 'programmers-just-play-the-game' week, serving lunch and dinner to encourage playtime during breaks and creating special game builds for employees to take home and play in their spare time. These initiatives were put in place to increase overall time spent with the game. Participation, of course, was voluntary and all time with the game was tracked using the telemetry system.

Using the time and gameplay data tracked, we created playtime leader boards, daily, weekly and monthly achievements and held contests with prizes to incite competition between players, see Fig. 7.8. By adding content specific achievements, we were able to direct testing towards areas of the game in need of additional attention.

In order to keep the game playable for everyone at any point during development, issues blocking progress through the game (such as broken quests or unavoidable crashes) had to be treated with the highest priority. In addition to the normal build validation procedures, we added a bot-driven gameplay 'smoke-test' as part of the hourly build process that covered the basic game interactions (movement, object interaction, combat, etc.). A failure by the automated bot to complete the post-build test level would automatically fail the build and protect users from updating to a defective game.

The efforts were very successful – too successful in fact. By the end of the project, more than 1,100,000 games were played on more than 1,100 different machines. The volume of issue reports filed per day (on average more than 50 per day) was overwhelming our ability to validate and triage the problems to the right


Developer Achievements

This page lists your developer achievements. You can filter the list using the dropdown below.

All Achievements (171)

	Title	Requirements	Status	Date
	A-Head Of The Pack	At the end of a week (Sunday), be the person who logged the most hours across all modules within your department. (1 hour minimum)	achieved	09/01/05
	Bug Hunter	File at least 20 bugs / feedback items through the My Feedback Interface.	achieved	08/09/22
	Ceremonialist	Ceremonialist - Complete Urn of Sacred Ashes (Side against Cult)	achieved	09/04/04
	Chanter's Friend	Complete all of the Chanter board quests except for Allison's quest. Reloads are OK, no cheats.	achieved	09/03/15

Display User:
Ray



- My Profile
- My Played Games
- My Game Options
- My Feedback
- My Achievements
- My Screenshots

Options

- Achievement History
- Achievements By User
- Link To This Page

Fig. 7.8 BioWare's telemetry backed developer achievement homepage (Used with permission from Bioware)

people on the project. Additionally the quality of issue reports from voluntary testers was generally low. These users, often playing in their spare time, were more than willing to report the problem they encountered, but didn't feel like spending time to fill out all the fields in the issue tracking software. With important information, such as 'reproduction steps' and 'build number' left empty, many issue reports in the system became borderline useless and required additional follow up from QA to salvage. We tried to correct this problem by making the fields in question mandatory, but that quickly led to a steep drop in reported issues, along with all kinds of creative values polluting those mandatory database fields (such as *Version found*: 'yes'). Thus, I would say that *the willingness for voluntary testers to report issues decreases with each mandatory field added to your issue tracking software*. To make things worse, most testers were not able to identify the correct recipient for issue reports, causing reported defects to sit around for days until they could be routed to the correct developer. Ultimately, we realized that our traditional issue tracking process was just not working for the size of the project and the number of voluntary testers involved.

Unwilling to give up on the increased quality assurance coverage provided by these testers, the decision was made to modify our issue tracking process instead. Specifically, we set out to create a new process and supporting software solving the following problems:

- Complexity
 - Reduce the number of mandatory fields on issue records as much as possible.
- Issue Quality
 - Improve the quality of issues field by voluntary testers.

Fig. 7.9 BioWare’s legacy issue tracker interface (Used with permission from Bioware)

- Duplicate Submissions
 - Reduce the volume of duplicate issue submissions.
- Issue Routing Problems
 - Get issues quickly to someone able to address them.

To enable voluntary testers to file defects effectively, the complexity of our issue tracking workflow and the filing interface had to be reduced to a point that even a non-developer (e.g. users from IT or Administration) would be able to submit issues. Below, I will discuss some of these issues in more detail.

7.2.3.1 The Issue Tracker Interface Problem

Filing a defect report on *Dragon Age: Origins* was too complicated. Our issue tracker exposed 32 fields to the standard user, ten of which were mandatory, see Fig. 7.9.

After analyzing our issue database, it was apparent that the mandatory fields were almost completely worthless:

- Any text field that allowed free form entry (e.g. Version Number) was polluted with typos, random letter combinations and text in different formats, making it completely useless.
- Most fields with dropdown style list options had grown to a point where most users were unsure which value was applicable for a given situation.
- A large number of fields contained a rather unlikely number of default value entries, indicating that most users just ignored the field completely.

File a Bug

Please select an appropriate category and describe your issue. The first line of your description will also become the bug title. Bugs filed while the game is running will be amended with available technical information such as environment and build.

The screenshot shows a web form titled "File a Bug". At the top, there is a "Required Fields" section with a dropdown menu for category selection. The dropdown is open, showing options: General, Client Programming, Art Issues, World Design Issues, Server Programming, Combat, Gameplay, GUI, and Ability Shakedown. The "General" option is selected. To the right of the dropdown is a "TO:" field containing the name "Georg Zoeller". Below the dropdown is a large text area for the bug description. At the bottom of the required fields section is a checkbox labeled "Generate Email Notification" and a set of standard window control icons (minimize, maximize, close). Below this is an "Optional Fields" section with a header "Attach Screenshots and Savegames". At the very bottom of the form is a "Submit This Bug" button.

Fig. 7.10 The new, telemetry enabled interface (reduced to four fields only) for reporting bugs (Used with permission from Bioware)

Similar observations could be made regarding optional fields. Most of them were empty or at default value or had developed different meanings for different teams, creating confusion whenever issues were passed between teams.

Starting from there, we decided to radically simplify the issue tracking interface used by non-QA players. Instead of relying on the user to fill fields such as ‘Build Number Found’, ‘Platform’, ‘Branch’ or ‘Game Area’, the new interface queried our gameplay telemetry system for these values in real time when the user decided to fill a bug, see Fig. 7.10. Through integration with EA’s Active Directory, the issue reporting website automatically detected the active user’s running game and pre-filled all necessary values – no user interaction was necessary. All in all, we managed to reduce the number of total fields on the main issue tracker interface to five, none of them mandatory.

7.2.3.2 Removal of Systemic Issues from the Issue Reporting Process

Warning pop-ups, recoverable script error messages and asserts had always been a problem on the project (Fig. 7.11). Usually these issues would appear in a form of a standard windows message box and almost every user would ignore them. If they couldn’t be ignored (such as in case of a crash or code assert), hundreds of these issues would be reported without any useful context.

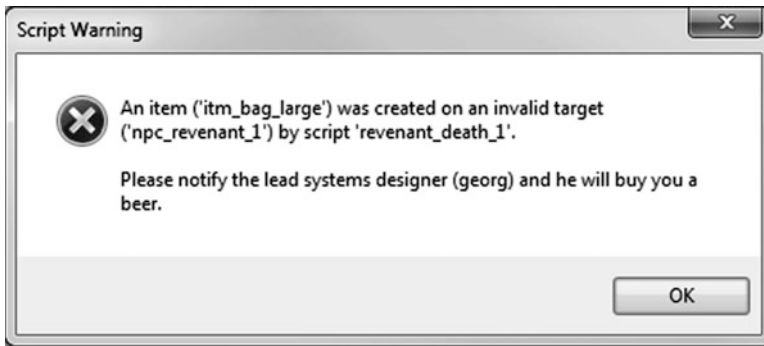


Fig. 7.11 Sadly, even incentives coupled warning messages were usually ignored (Used with permission from Bioware)

Leveraging our telemetry system, we removed any kind of visible ‘popup’ warning and recoverable error message from the game. Instead, these code- and script-warnings, along with asserts and crashes were captured and sent to a global database maintained by a QA crash engineer charged with tracking them. Detailed machine information, game state and call stack for each reported incident were automatically supplied to help with the investigation process. We also added the ability for developers to subscribe/own the warnings they added to the code. This allowed them to receive aggregated report emails for their warnings and asserts.

Some of the most useful events in this context were:

- Memory Pool overruns: a machine overflowing the allocated memory pools for textures, audio, etc.
- Game Systems Failures: instances of failure in game systems such as pathfinding, player movement, etc.
- Asserts and Crashes: Any kind of developer defined assert condition and crashes, along with memory dump and call stack for later processing.
- Resource Failures: attempts to load or create a missing asset in game.
- Performance Violations: machine exceeding load time goals or experiencing prolonged frame rate drops.

Removing these problem reports from our issue tracking database significantly reduced the volume of duplicate or useless records.

In addition to enforcing accuracy for many of the previously mandatory issue tracker fields, the tight integration with the telemetry system allowed us to further improve the quality of individual issues and the ease of reporting them. Since the telemetry system tracked machine hardware and software information for each start of the game (see Fig. 7.12 for an example of hardware information), storing it along with issue reports was a no-brainer. It proved most useful in identifying cases of elusive hardware and software problems, such as out of date drivers or known problems with certain types of CPUs/GPUs. We were then able to tie specific crashes and memory corruption issues to individual machines in the building that either turned out to be defective, or were running exotic hardware not present anywhere else in the building.

MAC Address	00-1e-c9-59-f1-0c
Hostname	rob.bioware.ad.ea.com
IP Address	10.0.15.145
RAM (GB)	1.01
CPU	intel(r) core(tm)2 duo cpu e6750 @ 2.66ghz
CPU Count	2
CPU Clockrate	2659
GPU	nvidia geforce 8800 gts 512
GPU VRAM (MB)	0
DirectX Version	4.09.00.0904
Screen Width	1024
Screen Height	768
Aspect Ratio	1.333
Timestamp	10/2/2009 1:17:22 PM

Fig. 7.12 Hardware specifications of a user (Used with permission from Bioware)

With the previous bug tracker interface, users trying to report issues with art had to manually take screenshots in game, convert and upload them to the issue tracking website. The same manual process applied to attaching savegames. The new interface not only exposed screenshot functionality through simple one-click controls, it also enabled various debug related displays in game based on the category the user selected for the issue. For example, selecting ‘Pathfinding problem’ as category would automatically signal the game to enable visualization of the A* pathfinding grid before taking the screenshot.

Due to the telemetry system’s awareness of the game’s build number and platform, we were also able to solve the problem of version and platform incompatibilities between savegames by providing ‘one click’ load right from the filing interface that would not display outdated or incompatible savegames. In addition, the availability of user’s exact location in game at the time of issue report, allowed us to add the in-game coordinates to every bug along with a simple ‘teleport to location’ feature on the main issue tracker interface. This proved to be an invaluable addition as it enabled location specific search and reports, visualization of issues in game. Hence, designers and developers can investigate issues by space; in other words, they can investigate questions like *show me issues that are around this location?* We later extended this system with detailed information about the player’s in game position, camera facing and visual settings along with every screenshot, which enabled the recipient, usually an artist or tester, to recreate the scene presented at the press of a button, which is a crucial feature for developers during the debugging phase.

Additionally, we tracked every tester’s in-game movement based on player input and time (~20 samples per second). Thus, enabling Quality Assurance management to develop ‘path coverage’ maps for each area of the game and ensure that every inch of the game world was covered by testers for every milestone. Furthermore, with data about almost every player interaction within the game, we were able to automatically provide basic reproduction steps with many issues. Limited to issue reports of certain

categories (e.g. Quest Problems), for every issue report, the system inserts a link to a full list of player interactions with any quest relevant objects in the game.

7.2.3.3 Detecting Duplicate Submissions

With several hundred players reporting issues, the volume of duplicate issue reports in our database had increased dramatically. Thankfully, the new telemetry backed issue database enabled us to deal with this problem in several ways. With in-game locations attached to every issue report, we created a new report that dynamically listed open issue tickets around the player's current location in the game. This gave testers a quick way to check if a problem had already been reported and significantly cut down the number of duplicate reports submitted.

The report also enabled highly productive QA verification passes. Instead of individually jumping to each issue and weed out duplicate issue reports on the fly, testers were able to play through an area and validate open and fixed issues around them in one setting. By displaying issues grouped by area and coordinates, location specific duplicates such as missing textures now were easily detectable and could be closed out quickly by QA.

7.2.3.4 Issue Routing Problems

Allowing a large number of users, not necessarily familiar with the full project hierarchy, to report defects created some challenges in triaging issues to the proper recipients. Often the problems ended up assigned to the wrong person for days, in some cases weeks before reaching the proper team. The new interface allowed users to report issues without defining a recipient. In those cases, we would generate a default recipient based on issue category (Art, Design, etc.) and weighted text analysis. For example, by selecting Category: Art and mentioning 'Visual Effects' inside the issue text, the system would automatically suggest sending the problem to the lead visual effects artist for further triage. Any issue that could not be resolved was sent to a dedicated QA resource for triage. Additionally, by querying the telemetry logs and vacation tracker software (the component of the calendar system we use to keep track of when people is on vacation or with time off), the new issue tracker would also prevent issues to be assigned to developers on vacation or time off and instead transparently reroute them to the appropriate team lead for manual triage, along with email notification. While not 100% effective, the measures taken to improve issue routing allowed us to stay on top of the constant flow of new reported issues for the duration of the project.

7.2.3.5 Known Crash Inventory

Automatic crash and assert tracking helped replace our previous, less than scientific methods of determining stability. Thus, instead of relying on the ability of QA

Callstacks for a specific build Other 'Build Integrity' reports... [back to report lists]

All callstacks for build 9635

RunId	TopOfStack	CallstackHash	Time	RunTimes	User
1039061338428064	embeddedstorage::idle	1ed75d76c36573b138f5b-c02873a5a1864dfcb9	8/26/2009 12:06:53 PM	6.15	unknown@360user
1156834686120	embeddedstorage::idle	1ed75d76c36573b138f5b-c02873a5a1864dfcb9	8/24/2009 10:03:26 AM	5.57	mlang@bioware.ad.ea.com
68579393132272	embeddedstorage::idle	1ed75d76c36573b138f5b-c02873a5a1864dfcb9	8/24/2009 10:00:00 AM	3.43	mlang@bioware.ad.ea.com
1083913241463	ccoreature::loadfromtemplate	255e5406aa88574d7a38cddaeec133552ee4cb66	8/21/2009 10:04:52 AM	6.53	dclifford@bioware.ad.ea.com

Export Grid To Excel Subscribe to this Report

Fig. 7.13 Sample ‘crash report’ for Dragon Age: Origins (Used with permission from Bioware)

testers to reproduce a reported crash, the new system was able to reproduce the play through automatically using telemetry establishing an inventory of known crashes (see Fig. 7.13).

To maximize the value of the new crash inventory, we developed a powerful ‘crash center’ application that automated the process of unwinding stack traces from all platforms to generate a crash report with unique signature for each incident. This enabled tracking of individual crashes across builds of the game and gave valuable information, such as date of first and last occurrence, frequency, affected platform, in-game locations and more to the engineering department. Even better, we are now able to detect regression in the game code – if a crash signature previously marked as fixed reemerged on the tracker from a newer executable, the associated issue report would be automatically reopened as a critical issue.

By cross referencing collected hardware and software information with the crash database, we can dramatically cut down the investigation time into crashes, often identifying the single machine in the building on which a particular crash would happen (e.g. due to faulty memory). In an effort to cut down on the number of known, resolved issues, the crash center application was able to disable older builds of the game by flagging them as defective. Users registering on the telemetry tracker with such an outdated executable were informed that their build had known issues and had been disabled, prompting them to update.

7.2.3.6 Automated Performance Validation

Another highly productive use for the telemetry system was automated performance validation. With Quality Assurance testers spending the majority of their time testing design and art related content, we had some deficit in technical testing, especially for game performance (frame rate, etc.). Capturing performance related information from players during regular gameplay had mixed results. Without being able to control or measure the impact of external factors, such as expensive editor software concurrently running with the game, automatically captured performance information from the game turned out to be imprecise. We resolved this by developing a set of automated tests running in a standardized, controlled environment.

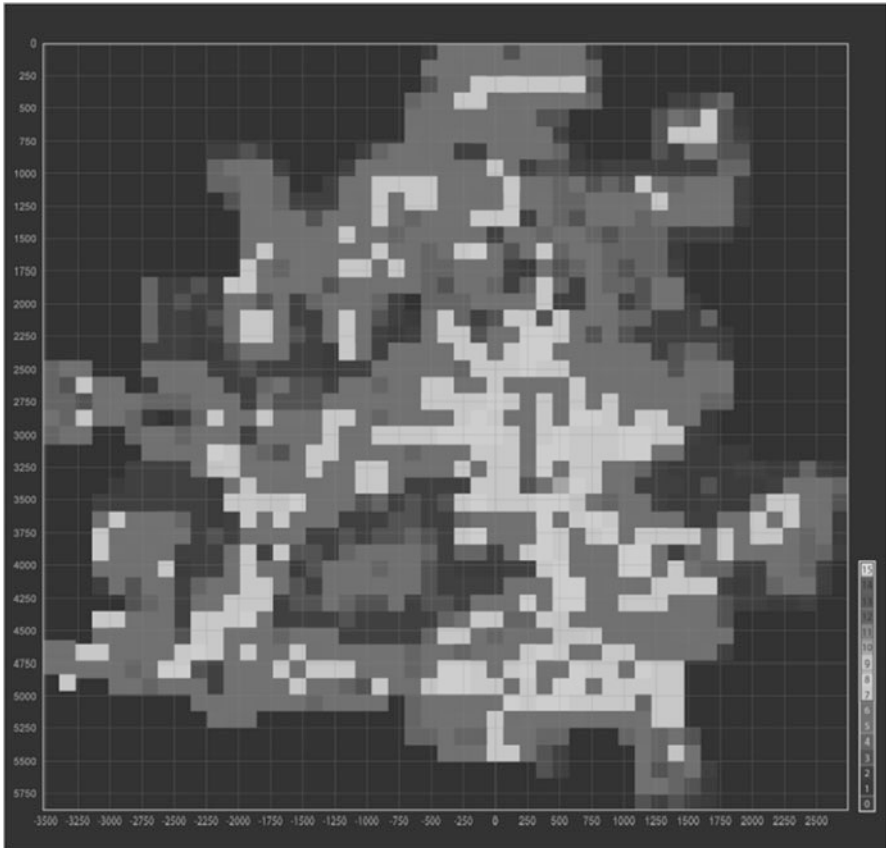


Fig. 7.14 Sample ‘Terrain texture density heatmap’ (Used with permission from Bioware)

As part of the test, a set of automated players (bots) would traverse the game nightly and sample important performance metrics such as memory usage, frame rate, draw calls, etc., on a highly detailed grid.

The data was propagated to the telemetry system and presented in a set of reports, updated daily. Significant changes between nightly tests automatically triggered email alerts to the relevant stakeholders, effectively preventing performance related issues from creeping into the game.

Heatmaps make for an especially effective form of presentation of this kind of data, as they can easily be compared and day-to-day changes appear very obvious to the person analyzing them. Figure 7.14 shows an example of a heatmap of terrain texture density in a level, allowing artists to quickly determine locations where significant optimizations through texture flattening can be made (see Chaps. 17, 18, and 19 for more on telemetry analytics and visualizations).

7.3 Lessons Learned

Ultimately, the telemetry backed issue tracking process was successful beyond our expectations. With many BioWare employees across several projects playing the game and sending feedback, we increased the testing coverage for the project by tens of thousands of hours. The simplified reporting process allowed even a person not familiar with the project to file high quality defect reports without any technical background knowledge, which lead to a dramatic increase number of reported, actionable issues.

Beyond simple issue reports, the metrics gathered from individual games helped us understanding the context of player feedback received:

- With the overall amount of time spent in game available for each player, it was easier to decide the weight of subjective, critical feedback.
- We could now determine how much time each player spent within the game and solicit additional feedback.
- It was now possible for us to ‘debug’ issue reports, e.g., a player reporting falling out of the world could be investigated by reviewing his in-game movement records and locating the hole in the level geometry that allowed him to escape the valid game area.

Finally, we gained the invaluable ability to assess the performance of external quality assurance services. With detailed metrics on the time spent in game, ground covered as well as quantity and quality of submitted feedback, it is easy to predict the time needed to test each part of the game and to identify highly productive testers for retention.

7.3.1 Metrics at BioWare, Past *Dragon Age: Origins*

Since *Dragon Age: Origins*, the use of developer-facing telemetry as well as end-user focused telemetry during development and post product release has become mandatory. Instead of a few developers handling the topic in their spare time and in addition to their regular duties, the company now employs a full analytics department servicing several studios across the BioWare family. For the development of *Star Wars: The Old Republic*, we deployed metrics to an unprecedented level during development, gathering information on player behavior from closed and open testing and using it to iterate on every aspect of the game. The scope of the project – a massively multiplayer game with hundreds of thousands of testers and millions of potential players – forced us to completely redefine our approach for collecting, storing and analyzing data. Instead of simple relational databases we are now using large scale data warehouse solutions and enterprise level analytics software to make sense of the terabytes of data generated by our users. Data presentation also had to change. While information on individual users still has value, data aggregated from

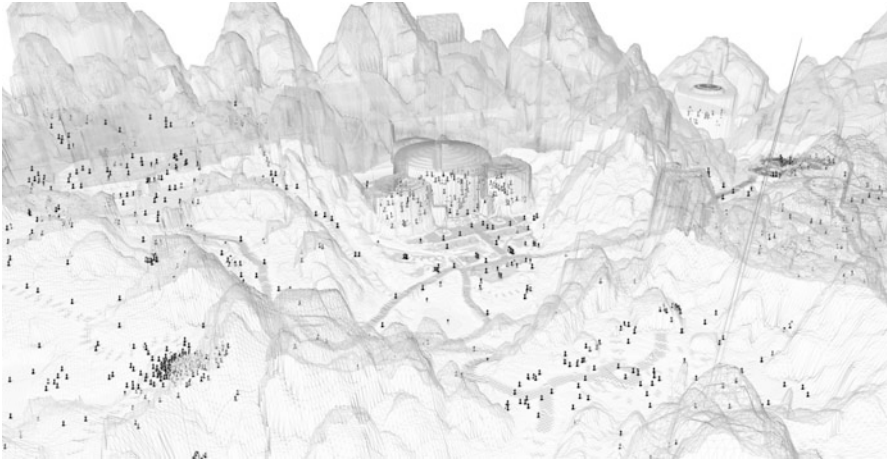


Fig. 7.15 Sample 3D wireframe visualization of game data (area geometry, creature locations) (Used with permission from Bioware)

hundreds of thousands of testers requires a different level of visualization than data from a few hundred players. In addition, a combination of proprietary tools and customized enterprise software, such as *Tableau* has replaced our homegrown web-frontend from previous games.

To deal with the massive amount of game content and its impact on user behavior, we added a new technique to our data analysis repertoire: Data Mashups. Using flexible, proprietary software called *Spatial Visualization Toolkit*,⁵ we are able to project both game content and asset data (such as placed creatures, trees, rocks, buildings, traps, etc.) and user behavioral data (such as movement patterns, chat, combat outcomes and player to player interactions) onto two dimensional and three dimensional representations of the game world, allowing designers, artists and data analysts to gauge the impact of game content on user behavior (see for example Fig. 7.17). This approach allowed us to understand and validate the impact of newly introduced features (for example ‘Fog of War’ – a layer of opaque fog that hides areas on the game’s map until they have been explored by the user) by comparing the user behavior before and after introduction of the feature.

Figure 7.15 demonstrates spatial mapping of player created data, in this case high frequency keywords taken from in game chat, onto the map of a game level. We use these ‘chat maps’ to determine sources of confusion and player questions in early game levels (Figs. 7.16 and 7.17).

For us, it seems clear that metrics, during development and post release, are now an integral part of game production and that the quest to handle the ever increasing complexity of game development cannot be won without them.

⁵<http://gdc.gulbsoft.org/2011-gdc-online-talk>

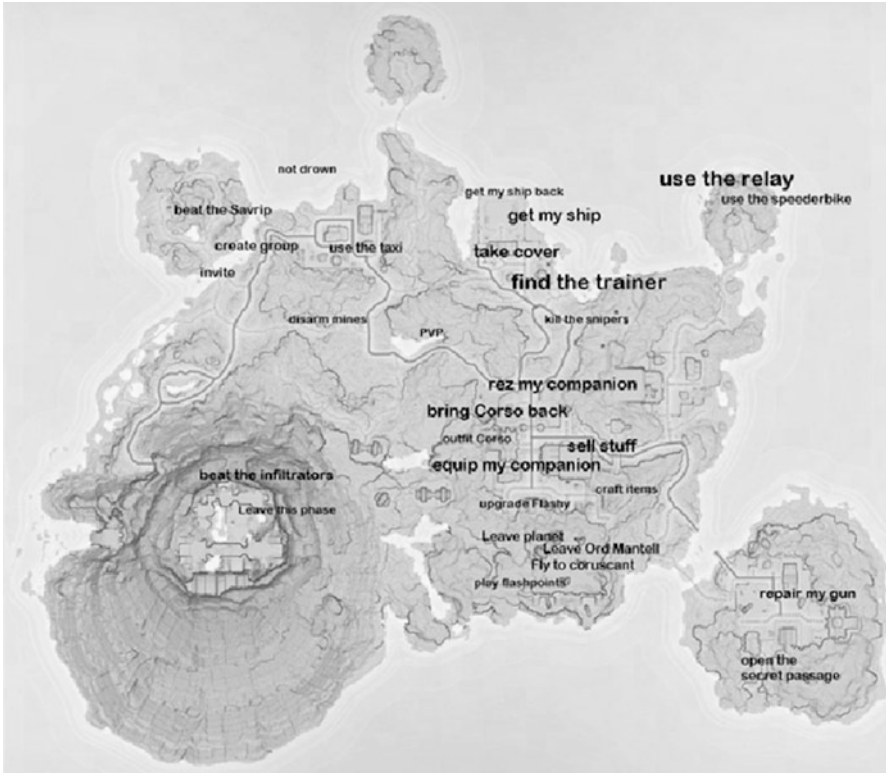


Fig. 7.16 Player ‘most common chat questions’ projection on Game Map (Star Wars: The Old Republic) (Used with permission from Bioware)

7.4 Conclusion and Summary

When we initially started to gather metrics from the game during development, we had no idea that it would ultimately lead us to change many of our traditional development processes. Gradually, over the course of several years, the data collected from developers and testers opened up more and more opportunities for us to optimize workflows, understand behaviors and make better games. However, in addition to the positives, there were some negatives. We encountered a number of pitfalls and findings over the course of the project that I will uncover in the next subsections.

7.4.1 *Bad Metrics Can Cause Real Damage*

After working with data for a while, it becomes incredibly easy to trust it blindly. Inevitably mistakes will happen when adding new metrics hooks for the game or

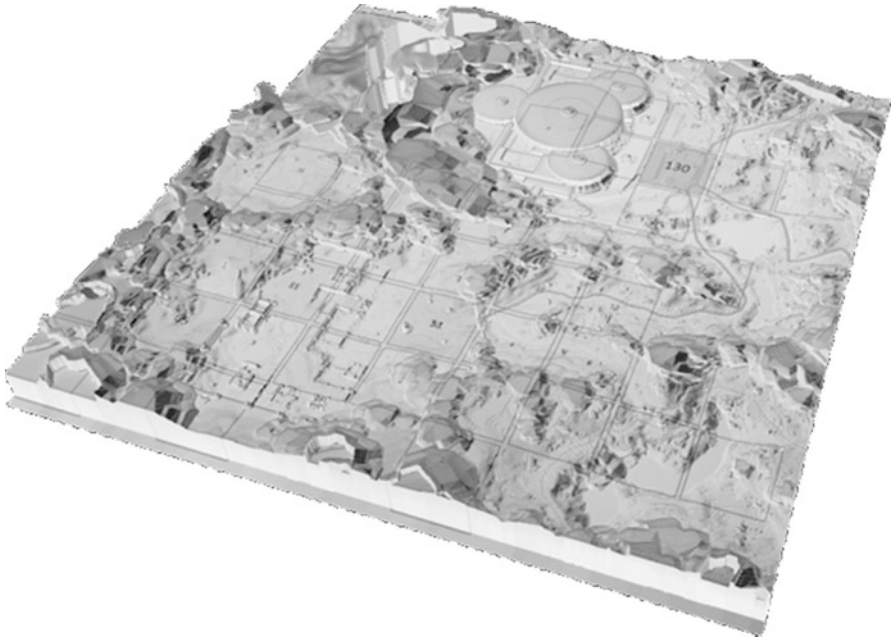


Fig. 7.17 Projection of user events (intensity/numbers – in this case character death) on top of game topology (Used with permission from Bioware)

when old code is changed without informing stakeholders that their data will also change along with it. On one occasion, we triggered a large, multi-day investigation into degraded game performance. We then realized that the values communicated to the telemetry system had all been changed by a factor of 10 due to a bug. There are plenty of opportunities to waste time investigating phantom issues caused by bad data, so being critical of data when it doesn't appear to make sense is a good thing to do.

Lesson Learned: Always test your hooks if you plan to make decisions based on the data. Important hooks need to be retested frequently. When in doubt, don't trust the data; validate the hook in game first.

7.4.2 Too Many Custom Proprietary Tools

Since every part of our internal metrics system was homegrown, we ended up with a lot of proprietary code that required constant maintenance and upgrading. For example, the deep integration of our metrics frontend with Electronic Art's Active

Directory service, originally created to streamline access control and avoid manual user management, created massive headaches for us when trying to add outsourced Quality Assurance testers to the system. While it would certainly not have been possible for us to integrate an off the shelf solution with our internal workflows as effectively as we could integrate our proprietary, homebrew solution, we also failed to seize some opportunities to replace non-scaling parts of our system with externally supported software.

Lesson Learned: When a homegrown tool or service becomes wildly popular and used well beyond its original purpose, it becomes necessary to assign a dedicated resource for maintenance – or to replace it with an external, vendor supported stock solution when available.

7.4.3 *Game Developers Play Differently*

Another problem we encountered was the fact that developers play video games very differently. They are generally more inventive, easily work around problems and know who to contact to get their problems solved. This meant that analyzing design related data, such as game difficulty, produced highly misleading results. The game appeared to be easy and players seemed to have little trouble figuring out even the more complex game mechanics. Only later, after comparing metrics from external focus testers with the data captured internally, we realized how much patterns of in-game behavior found in developers and focus testers differed.

Lesson Learned: While it is tempting to analyze design and behavioral metrics taken from game developers, it is important to use outside focus testers to compare and validate results.

7.4.4 *List of Useful Development Telemetry Hooks*

The following is a short list of some of the more useful telemetry hooks we identified over the course of the project, along with the reason of why they are important.

- Game Start Timestamp/Game End Timestamp – Not only provides the overall time spent running the game client, but also context on when during the day people play, times they are unable to play and how long they are able to run the client on average.
- Idle Time – Through simple ‘idle/away from keyboard’ detection in the client, this information is necessary to discern how much time inside a game session was actually spent interacting with the game. This is very valuable when calculating

stability metrics for the game, as many clients sitting idle on a load screen, for example, could easily create the impression that the game is very stable, when in fact it could be prone to crashes during normal play behavior.

- **Game Saved/Game Loaded.** In order to understand player behavior in context of the game's story, understanding when a player reloads a saved game to continue a previously stored game session is vitally important. Without it, each game session would be seen as a stand-alone entity and the context of a player's progression through the game would be lost.
- **User Interface usage.** Information such as where and when a player opens the in game map interface allows designers to understand if a level or area is confusing to the player. This metric also surfaces which interfaces in the game a player spends the most time interacting with and can guide decisions to prioritize and streamline these interfaces during development.
- **Preference/Settings changes.** Understanding when people change away from the default settings of the game helps inform decisions on changing those defaults if needed. This can also provide information on questions such as 'which monitor resolution do most players/testers run?'
- **In world object interactions.** Tracking interactions with objects and creatures in the game world allows designers to understand which objects are never used (e.g. because they are defect). Combined with information about player movement, it also allows understanding the flow of player's through an area and whether or not players interact with the content in the way the designers expected. More than once we redesigned areas in order to maximize chances of players stumbling on an important piece of content that otherwise would have been ignored by a majority of players.

About the Author

Georg Zoeller worked as Principal Designer at BioWare and contributed to the release of *Neverwinter Nights: Hordes of the Underdark*, *Jade Empire*, *Mass Effect* and *Dragon Age: Origins* and *Star Wars: The Old Republic* since joining the industry in 2003. He relocated to Singapore mid 2012 to become Technical Creative Director for Ubisoft Singapore.

Chapter 8

Interview with Nicholas Francis and Thomas Hagen from Unity Technologies

Alessandro Canossa

Unity 3D is an integrated authoring tool for creating 3D interactive environments, such as games and real-time 3D visualizations. Unity consists of both an editor, for designing content, and a game engine, for executing the final product. Over the recent years since its inception in 2006, Unity was given several awards, including Wall Street Journal 2010 Technology Innovation Award and Gamasutra 2009 Top 5 Game Companies. Additionally, the company was held partly responsible for the democratization of the game development process.

This is an interview with two key figures behind the Unity engine. Nicholas Francis – Chief Creative Officer at Unity Technologies, responsible for product and feature design. Nicholas is one of the founders of Unity Technologies and has been working in the company for more than 10 years, before that he was just a dreamer conjuring ways to make game development a smooth, fun process. He has been credited as the grandfather of the Asset Store, just an example of Unity’s approach to problem solving: building tech so that people can sell stuff to each other, optimizing small teams’ resources and skills.

Thomas Hagen – He worked at EIDOS and Square Enix as Online Development Manager to build their metrics infrastructure for a number of years. He started writing software in the early 1980s at age 11; today he is an independent contractor and works with Unity Technologies to develop analytic systems.

Q: How do you think telemetry can be useful for the industry?

Nicholas: We at Unity Technologies believe that telemetry and metrics are going to be crucial for making better, more fun games and that is the core proposition of our business model. We are not interested in revenue optimization; we just want to help people make fun games.

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

Most of the time what we see is developers trying to minmax, micro-optimize and balance their designs, but when addressing metrics as a tool to achieve that goal, they acknowledge the relevance of the tool and at the same time their eyes wander, looking for someone else to talk to. I believe it is a human trait: when we don't know exactly how to do something, we will do anything else, procrastinating the blurry task indefinitely. Mobile game developers, unless they have a past as web developers, they have no idea how to set up a server, and that is enough to begin the cycle of procrastination. They would do the most amazing things in order to avoid doing something they don't perfectly understand. And that applies even more when open-ended thinking with no concrete question is involved, such as in exploratory analyses (see Chap. 12).

Q: What can Unity do to help in that regard?

Nicholas: I believe that most of the time some very simple analytics will get 90% of the job done. We want to deliver basic solutions that allow anybody, with no previous knowledge, to include simple analytics from the moment they begin authoring a game. Due to the fact that we provide a somewhat controlled environment, there is a ton of tools that we can provide out of the box, for example, tracking performance metrics such as frame rates, slowdowns, etc. We can also provide end-to-end hosting services integrated in the editor.

Another aspect is the presentation: in order to win over the initial hurdle and begin using analytics tools, we want to invest considerable resources in visualizing the data to provide an environment where developers can feel free to play around with their data. The key concepts are: simplicity of out-of-the-box analyses, accessibility of tools, and playfulness of visualizations. Complexity can be added via APIs where advanced developers can expand the basic solution. We tried it on our own skin: playing with analytics is addictive; we believe that developers will easily get hooked and will soon start toying around with the API.

Thomas: I expect the Unity Asset Store to be soon crowded with analytics extensions developed with the API.

Q: Do you think that a lack of clarity on potential benefits of analytics is also part of the problem for developers?

Nicholas: Not really; every developer knows that Zynga is racking in cash and everybody suspects that it has something to do with analytics. So there is a basic desire and positive bias towards it, but it is difficult to take it further than that. It is our intent to help developers turn raw metrics data into meaningful insights and actionable information.

Thomas: Game directors, CEOs and studio heads are very aware of the potential of the tool, they are even aware of the different costs and benefits for each type of stakeholder (see Chap. 3) The more business oriented you are, the more you require data to make decisions. However, if you come from the creative side of development, you are taught to trust in your vision. Creative directors and designers are more resistant to the idea of using metrics; they often feel challenged in their creative vision. It seems easier to convince top management, but at the same time it is impossible to deploy telemetry solutions without having the whole team on board.

A viable solution could be to remove any entry barrier for all levels of development: if for example level designers find analytic tools enabled and ready out of

the box, they will have no reason not to experiment with the tools, and they may eventually become advocates for the practice and help convincing colleagues. It is our vision to turn every step of game development into a game itself, and that applies to analytics as well.

Q: What are the metrics you plan to include initially as built-in features in the editor?

Nicholas: We expect all the variables currently tracked in the Profiler (CPU usage, rendering, memory, audio, animation, etc.) to be available as metrics for the developer. These include memory usage, CPU and GPU usage, streaming in and out of memory, etc. In addition to these basic performance processes, we already track automatically many variables from the cameras and the viewports, such as position relative to world coordinates, when players log in and log out, when players start a play session, when and which levels are loaded, when players stop a play session, and how long time passes between play sessions.

Thomas: We have two main target users: designers and coders. We want to provide a solution that caters to both: providing player metrics for the first group and performance metrics for the second group.

Q: Ideally, when do you see analytics becoming a part of the development process?

Nicholas: As it stands now usually developers devote resources to metrics collection and analytics very late in production, mostly at the beta stage. This is usually due to resource shortages. It is indeed difficult to put in place a system that tracks variables when there is no game at all. However, the appearance of metrics tracking that late in the process is a missed opportunity for both monitoring the evolution of software performance and stability, and charting the change of gameplay dynamics through iterations. We plan to change that by enabling out-of-the-box automatic tracking of several basic variables.

Thomas: We plan to develop solutions for rapid creation of heatmaps based on any game variable, both as traditional 2D visualization and as 3D particle clouds, integrated natively in the editor. Furthermore, we want to have a filter inspector to perform simple Boolean operations on sets of features to be visualized and compose basic spatial queries inspired by Geographic Information Systems; a sort of complex query compositor.

We might also go the Apple way and offer a very limited number of basic queries that can be applied in most cases, and later in time expand the offering with additional queries. This fits the development philosophy at Unity: ship few features very polished and expand constantly. Approaching the problem in this manner softens a potentially daunting learning curve, minimizes risks and facilitates early adoption by beginners.

Q: How many of your customers, which are relatively small in size, have user research departments? How will your tool benefit them?

Thomas: Only the most experienced game director understands and values the benefits of user research to the point of insisting to have a specialist on board. Otherwise, most game directors tend to enjoy being left on their own taking drastic, risky decisions about their games. A second factor is the size of the development teams: only medium to large teams have enough stakeholders, who may be interested

in keeping an eye on player behavior; in smaller teams, say less than ten people, only the lead designer pays attention to player experience, and often functions as a user researcher. No matter what, even in the best case scenario, user researchers have to agree with the game director on the variables to track. Usually, it is the game director who gives the final authorization to programmers to enable hooks in the code to tracking certain variables. Right now most of our customers are studios with a staff of around five people, and thus do not have dedicated analysts, but I see the situation changing soon: a streamlined environment instantly empowers even one-man teams to adopt analytics practices.

Nicholas: I believe that our tool, ideally, would empower user researchers to open the game in the editor, define variables and start tracking independently from the wishes of the game director. User researchers no longer have to depend on the game director greenlighting the tracking of specific variables; user researchers can decide what to track and implement it by themselves, at least during development; what to track from a finished, shipped game is still a big decision that should be agreed upon by all stakeholders involved.

Until now, due to the extra burden put on programmers and the fact that user research departments do not usually include technical people, it has not been possible for user researchers to have that kind of independence, but if our tool is easy enough to use, that scenario could change allowing user researchers, who have some or no technical expertise, to constantly tweak variables and metrics almost at a whim. This will also allow user researchers to seamlessly setup AB testing (see Chap. 4) both before and after release. They may also setup dynamic metrics, maybe not in the first iteration of the tool. However, it is definitely our dream to enable users to define and change metrics even after the game is deployed.

Q: Why do you think metrics collection and analysis seems to be in the Zeitgeist?

Thomas: Telemetry is currently perceived as a difference-making element. Due to the extreme competitiveness of the game industry, it is becoming more and more important. Unity has democratized games' development: the tools are affordable and the necessary skills fairly easy to obtain, so much so that games, like *Angry Birds* (Chillingo/Rovio Entertainment, 2009), can be created by a small team of students in a very short time. Creating successful games is no longer just a technical achievement, it becomes vital to know how to tweak and polish your creation to make it different from your competitors. And that's Zynga's real strength; most of the employees are engineers, some are from marketing, but almost no designers. They are not doing anything revolutionary in terms of design. However, whatever they are doing, they are doing it really well and they know precisely what their players are doing at any given time.

Nicholas: Another key factor is how much people are getting used to the concept of analytics. Taking any introductory web design class will expose you to the basic principles of Google Analytics, something simple enough for anybody to grasp and pervasive enough. People are getting comfortable with the concept of hits, and care enough about that to check more than once a day. Furthermore in pretty much every aspect of the business, it is necessary to justify spending and decisions need to be

backed up by facts. This brings analytics to the spotlight as it is a fast and cheap way to obtain facts. I will not be surprised if, within 5 years, game data analytics becomes an integral part of game education curricula.

Q: What are the main benefits of telemetry and analytics for the game industry?

Thomas: In order to be competitive, it is crucial to know how players are using your game. Obviously this fact translates differently for each of the stakeholders. Designers benefit the most during development, while management and corporate personnel are more interested in data acquired after release. For example, at Square Enix metrics are used to assess and spot piracy – *Tomb Raider: Underworld* (Eidos Interactive, 2008) was played on the Xbox by unauthorized accounts even before the official release. Using metrics, we were able to identify these users and enquired about how they received such early access. Another extremely important layer that can benefit greatly from metrics is the community of players. As a developer it is important to reward players that generate precious data for our production, something beyond leaderboards, like replays, smart match-making or gamer lifecycle information.

Q: Any suggestions to green developers that are interested in including telemetry in their games?

Nicholas: Use Google Analytics: it's completely free, rather simple to use, and, after only 1 day writing code, the rewards are immense. Additionally, the practice is invaluable because doing that can train anybody to the frame of mind necessary allowing them to start tracking data for games. It's a great testing ground before deploying more complex solutions. We are actually using it to track how our customers use the editor to develop their games. For example, we know that in 2011, 30 man-years were spent by our developers waiting for the engine to bake lightmaps. If we had a faster solution, we could have saved a person's professional life.

Thomas: Even developers adopting game telemetry solutions seem to make use of it mostly during production; as soon as the game is shipped they tend to lose interest. However, one thing that is missing from this practice and is sorely needed, is to continue analyzing player behavior data to verify if the design decisions taken during development and based on gut feelings turned out to be accurate guesses or complete misses. This might not bring immediate benefit to the game in question, but it can provide lessons for designers to mature their practice and craft.

Q: Do you think it is possible to mature the current practices in place in the industry? If so, how?

Thomas: I think that the industry at large has not yet leveraged telemetry to its full potential. In the projects I have been involved in, metrics were never gathered from within the editor, so the work process of level designers, for example, has not benefited as much as it could have. The Unreal Development Kit has few features already, and it's a great start, but it's not enough. Ideally anybody could decide to track anything from any game, and be able to do it without involving any programmers, if the tools are developed and integrated well enough within the editor. Relating to what I said before about making better use of analytic practices post-launch of a game: another aspect to be improved could be to harness the passion and commitment demonstrated

by communities of players to investigate how people play the game either by administering questionnaires or employing machine learning, as it was done with the game *Tomb Raider: Underworld* (Eidos Interactive, 2008) where a team of researcher from IT University of Copenhagen classified the behavior of 1,365 players utilizing emergent self-organizing neural networks.¹ It would be precious information for the marketing and management departments when making decisions about future projects.

Nicholas: Recently we tried to hire a number of professional analysts from marketing, they may be proficient with SPSS, but with Google Analytics they can barely hit “print screen”. It is hard to qualify a new set of experts who are able to turn data into meaningful knowledge, and provide insights on the reasons behind a certain phenomenon. The end point for any type of analytic enquiry is being able to answer the “why” questions rather than the “what” questions. And in this case, both questions and answers are game-specific. Thus, we, as providers of tools, can enable developers, but cannot provide the answers, as their domain knowledge is essential.

Thomas & Nicholas: we completely agree on one vital outlook for applications of telemetry and behavior profiling and that is: adaptive and predictive game systems. The AI Director in *Left4Dead* (Electronic Arts/Valve, 2008), an automatic AI system that orchestrates difficulty and pacing based on player behavior, proved how this technique has barely scratched the surface of what is possible to achieve when a game system is aware of how it is being utilized by players.

Q: What areas can you identify for future research with game telemetry?

Thomas: If we at Unity manage to identify and standardize the measurement of several game variables across games, such as defining a universal “player death” tag that works on all games where players can die, we can then start looking at differences between games and across different genres. This development together with a permanent player identity could potentially generate a gold mine of data, and analysts can start looking at a long term temporal dimension of play behaviors across different games.

About the Author

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and the Royal Danish Academy of Fine Arts, Schools of Architecture, Design and Conservation. His doctoral research was carried out in collaboration with IO Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches. His work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he maintains an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio.

¹ Drachen, A., Canossa, A., & Yannakakis, G. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of IEEE Computational Intelligence in Games (CIG) 2009* (pp. 1–8). Milan: IEEE Publishers.

Chapter 9

Sampling for Game User Research

Anders Drachen, André Gagné, and Magy Seif El-Nasr

Take Away Points:

1. Introduction to different sampling techniques used to select a subset of data and the process of applying them in game development.
2. Discussion of the pros, cons and challenges of using sampling to analyze player behavior.
3. Introduction to the core considerations underlying sampling theory and techniques.
4. Best Practices from the trenches of behavioral game analytics.

9.1 Introduction

At no point in history has there ever been so much data available on game players that can service game development as there is now. With the right tools and channels, developers can obtain detailed information on player behavior across the entire

A. Drachen, Ph.D. (✉)

PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark

e-mail: andersdrachen@gmail.com

A. Gagné

THQ, Agoura Road, Agoura Hills, CA 91301, USA

e-mail: gagne.andre@gmail.com

M. Seif El-Nasr, Ph.D.

PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

population of players, thanks to tracking software (DeRosa 2007; Kennerly 2003; Zoeller 2011; King and Chen 2009). They can also obtain information on infrastructure performance and development processes (see Chaps. 2, 6, and 7). This information can in turn be used to inform decision-making processes from the smallest of design issues all the way up to strategic decision-making as emphasized in Chap. 3, and seen in examples discussed in various chapters in this book.

The ability to track multiple variables over time from players (telemetry) can lead to big datasets. Such datasets can be costly and slow to work with (Han et al. 2005). Sampling is a technique used to resolve this problem most often. Sampling is a process of selecting a random or representative subset of a dataset with the goal of being able to answer a question about the properties of the entire dataset (or population) based on analyzing a subset of the dataset. In other words, sampling provides a way of dealing with massive-scale (and smaller scale) telemetry datasets through selecting a subset – a sample. Analyzing data from a carefully selected subset, which contains the same inherent properties as the population they are drawn from, reduces the computational cost and makes the analysis process more efficient, enabling rapid iteration of analysis (Han et al. 2005).

When one would use sampling and what constitutes a ‘big’ dataset is somewhat a nebulous concept. In relation to game telemetry, a useful definition for big datasets is: datasets too big to fit into the rapid access memory of a strong PC. Thus, a dataset above 8–10 GB would be considered big. However, what is big is generally related to the specific case and the resources available to the company in question. For example, heatmaps drawn using GBs of telemetry data on death events from hundreds of thousands of play sessions in an online multi-player shooter are not unusual (see Chap. 14). In comparison, a small development team testing a vertical slice of an indie title might consider 100 MB of telemetry data from 10,000 players big. Irrespective, working with samples is desirable when the company has constrained resources.

In this chapter, we discuss an overview of the key issues in game telemetry sampling focusing on behavioral telemetry and its use in Game User Research (GUR) (see Chaps. 21 and 22 for more on GUR work, or Isbister and Schaffer 2008). Specifically, Sect. 9.2 provides an introduction to some basic challenges of working with sampling in game development, and divides the process of game development into two components: before launch and after launch, which is necessary due to population differences and the nature of the data collection setup. Section 9.3 details some practical lessons when dealing with sampling, covering topics such as sample size, validity and outliers. Section 9.4 contains an overview of some of the most commonly used sampling techniques in game development. Finally, Sect. 9.5 contains a handful of advice on best practices based on our personal experience working with telemetry sampling. Note that the chapter is not a thorough account of all the available knowledge on sampling or sampling techniques, but rather a starting place. Many of the concerns and sampling techniques described are not unique to situations involving game telemetry, but similar across most situations where sampling is applied in social science, communication and HCI research/areas.

9.2 Basic Considerations About Sampling

The goal of sampling is to obtain a collection of items that **represent** the population – or in the case of game telemetry specifically the **behavior** of the population, whether it is players or any part of the system architecture, e.g. game servers – as closely as possible with as few items in the sample as possible. In general, samples should reflect the characteristics of the dataset they are drawn from. In order to preserve the characteristics of the parent dataset in the sample, it is necessary to first obtain knowledge about the parent dataset. Selecting data without considering the composition of the target audience for a game, or choosing data from 1,000 players from a dataset of 1,000,000 without understanding the properties of the parent dataset, will not provide samples that mimic the properties of the population of people or behaviors that the analysis is trying to uncover.

Sampling is a process that consists of a few generic steps:

1. Defining the population or dataset of concern.

We define a *population* as all the people (identified as unique accounts), items or processes that we want to investigate something about. As we are focusing on GUR, the population will generally be all the potential buyers or players of a game. A population could, however, also be all the servers on which an online game is run, or anything else we want to know something about, of which there are too many for it to be effective to work with all the units/individuals.

A common problem with some forms of populations, including “potential buyers of our game”, is that it is not possible to identify all the individuals/items in the population. Therefore, we cannot perform probability sampling (see Sect. 9.4), and cannot apply statistical tests to evaluate the likelihood of a pattern occurring in the sample also occurring in the population. A somewhat partial solution to this problem is to define a sampling frame where we can identify every person or item in the population. For example, if we are developing a game for the Danish market, for males above 18 years old, we might not be able to obtain a list of every single male in Denmark belonging to this population; however, we may be able to obtain a Danish national telephone directory, which contains most adult males in Denmark. This kind of population subset is called a *sample frame*, and is the common situation when performing pre-launch GUR work. The most important feature of a sampling frame is that it must be representative of the population. If it is not, sampling bias may occur, or worse, a sample may be unrepresentative. Typical reasons for bias occurring are: (1) the frame is missing some members of the population, (2) the frame includes non-members of the population, (3) members are included more than once (duplication), or (4) the frame lists clusters rather than elements (Lohr 2009).

We define a *dataset* as data that we have already collected and stored in an accessible form. The theory and methods regarding sampling is similar irrespective of whether we are working with sampling from populations or datasets; however, the situations where either is present varies and importantly, when we sample

from datasets (draw a subset) we have complete knowledge of the population's properties, we can check our results against the population (parent dataset) and can include any of the items in the dataset in the sample.

2. Deciding on a sampling strategy.

There are numerous methods available for sampling, and the choice of which method to use depends on a number of factors, including the available knowledge about the population/dataset, resources available, the desired accuracy needed, etc. There are two groups of sampling strategies: probability sampling and non-probability sampling. These are discussed in Sect. 9.4.

3. Determining the sample size.

Sample size is important because if a sample is too small, any patterns observed in the sampled data will not tell us anything about the population or dataset – and usually this is what we want to be able to do (e.g. will players in our target audience find the game fun to play?). Determining sample size is a process of choosing the number of data points needed. The number depends on the size of the population/dataset, and the desired statistical power. Sample size is discussed in more detail below.

4. Drawing the sample.

Finally, the dataset is drawn from a dataset or data collected from a population and the analysis process initiated.

9.2.1 *Sample Size Determination*

Analysis is conducted for a variety of reasons, e.g. to establish the similarity or difference between two groups of players, or to estimate a quantity in a population/dataset of players. Regardless of the type of analysis, estimating the minimal appropriate sample size is important – not only in terms of ensuring that a result obtained is useful, but also to avoid wasting resources on using samples that are too large.

Determining the minimum sample size in a situation is a challenge that has received considerable attention in statistics. A major component of probability sampling focuses on reducing the error margin for any given measure, which is strongly affected by the size (and properties) of the sample. While sometimes necessary, using small sample sizes can lead to wide confidence intervals or errors in statistical hypothesis testing (Bartlett et al. 2001; Fowler 2001).

The essence of this problem is that if we draw only very small samples, the risk of the sample not being representative of the entire population/dataset is substantial. Imagine for example sampling ten players randomly from a one million player dataset – the risk that those ten selected players are outliers, or coming only from the lower or upper part of the frequency distribution is fairly high. With, however, 1,000 players sampled, the statistical chance of these being distributed in a likeness of the main one million dataset is a lot bigger (see Fig. 9.1).

What this means is that generally size impacts sample representativeness in the most obvious and direct way: how well the sample reflects the properties of the

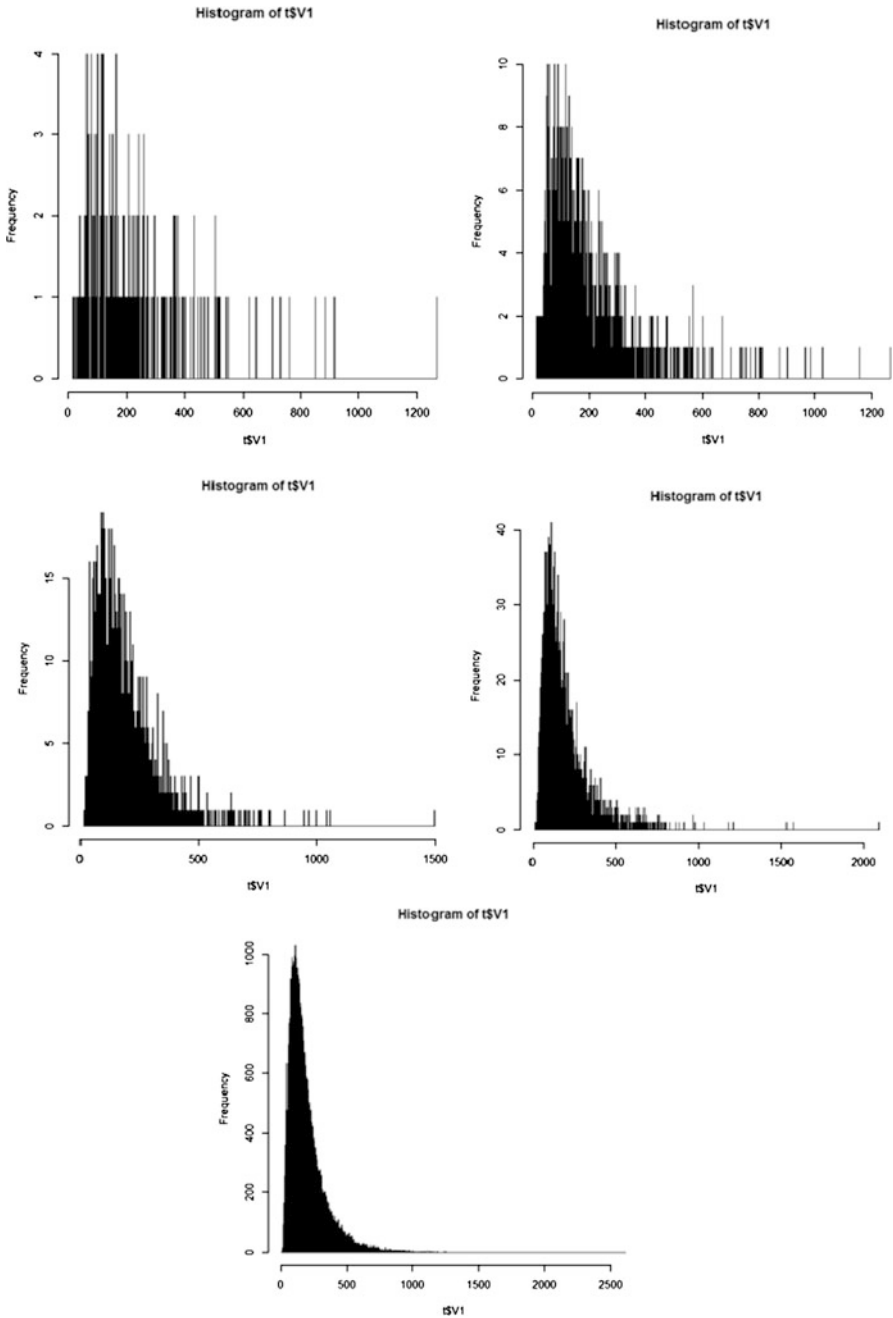


Fig. 9.1 Sample sizes and variance captured: *top left*: a random sample of 300 items. *Top right*: a random sample of 1,200 items. *Mid left*: a random sample of 1,500 items. *Mid right*: a random sample of 3,000 items. *Bottom*: a random sample of 100,000 items. Histograms use 1,000 bins. Samples progressively resemble the parent population with sample size. 1,500 samples give a decent model of the overall population, and moving from 3,000 to 100,000 data points in the sample does not improve accuracy substantially – given the kinds of questions usually asked of game telemetry data (Note that e.g. in engineering precision can be a very different beast than in player behavior analysis)

population/dataset. Simplifying a bit, larger samples will contain a bigger portion of the population, leading to increased precision (how the sample is selected also impacts its representativeness). This is due to two mathematical theorems: (1) the *central limit theorem*, which broadly states that given a set of conditions, the mean value of a sufficiently large number of independent random variables will be normally distributed, and (2) the *law of large numbers*, which states that the more times we perform an experiment, the closer the average is to the expected value, i.e. the average in the population/dataset (note that there are situations where increasing sample size will not substantially increase accuracy, or even not at all, for example if there are systematic errors in the dataset).

9.2.1.1 Factors Impacting on Sample Size Determination

There are a couple of important concepts to be aware of when determining sample size:

- **Type I and II errors:** Statistical tests are used to make inferences about a population/dataset based on samples (inferential statistics). For example, if we hypothesize that people who play the Mage class in our hypothetical MMORPG are mainly male, we could use a sample to derive an estimate – with a specific degree of significance – about all the players of the game.

There are two potential errors that can happen in this situation: (1) **Type I Error (α):** we believe that a given condition or pattern is present, when it is actually not present (e.g., we believe that people selecting the Mage class are also male, when there is actually no correlation between the two). Formally, we reject the null hypothesis (that there is no correlation between maleness and selecting the Mage class) although it is true. 0.05 is a commonly used value for the probability of a Type I error; (2) **Type II Error (β):** We fail to detect a given condition even though it is there (e.g., we fail to detect that males do select the Mage class significantly more than females). Formally, the null hypothesis is false, but accepted as true anyway. This can for example occur when there is a substantial degree of variation between samples, which is a common problem when investigating player behavior, especially when performing time-series sampling. 0.2 is a typical value for Type II errors (Cohen's $d=0.8$).

Type I and II errors are inversely related; meaning, if the threshold for accepting an effect as genuine is increased, we increase the probability that an effect that does not exist will be rejected. The exact nature of the relationship between Type I and II errors is not straight forward, however, because they rest on different assumptions. Type I errors assume there is no effect, whereas Type II error assumes there is an effect.

- **Effect size:** A statistical test can be significant (e.g. at the 95% significance level), without being important. Taking for example a hypothetical MMORPG, we may find a significant relationship between being male and selecting the Mage class, but the effect is weak – it explains only little of the variance in the dataset. Therefore, we want to measure the magnitude of an effect, known as effect size, which is simply a standardized and objective measure of the magnitude of an observed effect. Having calculated the effect size of a sample, the effect

size in the population can be estimated. Effect size can be calculated in different ways, one of the more common methods is the Pearson's correlation coefficient (Pearson 1900).

- **Statistical power:** The power of a hypothesis (or statistical test) is the probability that a given test will find an effect, assuming that one exists in the population of players. For example, in our hypothetical MMORPG, it may be that males have a tendency to play the Mage class. Power analysis is used to calculate the probability that a given effect size will be located by the statistical test applied to a sample of a specific size. An 80% chance is usually acceptable (Cohen's $d=0.8$), but this 4-to-1 weighting between Type II risk and Type I risk will not always be appropriate. For example, if we want to avoid false negatives (Type II errors), while accepting an increased chance of a Type I error. Conversely, power analysis can be used to calculate the minimum effect size that is likely to be detected given a specific sample size. Power equals one minus the probability of a Type II error occurring, i.e. a false negative.

The statistical power of a test is dependent on a number of factors, with effect size, the significance criterion (e.g. 0.05, 0.01 or 0.001) and the size of the sample being the most important. Increasing the significance criterion increases the power of a test, which reduces the chance of a Type II error, but increases the chance of a Type I error. Effect size directly impacts on power – the smaller the effect size, the bigger the power needed to detect the effect size. Sample size determines the amount of sampling error in a test result – in essence, effects are more difficult to detect in smaller samples, and thus increasing sample size is the easiest way to increase statistical power of a test – notably in a situation where we have behavioral telemetry from a large number of players. Finally, the precision with which data is measured impacts the statistical power, but this is usually not a problem with behavioral telemetry, where measures are precise.

Therefore, factors that affect sample size calculations include: **P value** (the statistical significance or probability value) the **statistical power** of the test being used and the **effect size** we attempt to detect. Firstly, a conservative cutoff for significance, i.e. using a small P value, will reduce the risk of interpreting a chance finding as being genuine (Type I error), but in practice requires a higher sample size in order to obtain. There are commonly defined values for P that are generally used, e.g. 0.05, 0.01 or 0.001. If we are doing probability sampling, the sample size is judged based on the confidence we can assign to a pattern in the sample being representative of the population/dataset. For example, we may want to be 95% confident ($P=0.05$) that a behavioral pattern we see in a sample of telemetry data is representative of all the players of a game. Secondly, having a high statistical power will make identifying any effects easier, but this is only achievable with a big enough sample. For power, there are also some typical values used, e.g. 80%. Thirdly, the size of the effect to be detected is harder to estimate the magnitude of. It is very important that a small effect requires a large sample and a large effect requires only a smaller sample. Erring on the side of caution and choosing a small sample is usually possible when dealing with a post-launch scenario where there are plenty of players in the dataset, but in a pre-launch scenario increasing sample size can be costly, and thus setting the expected effect size larger may be required, at the risk of ending up with an under-powered analysis.

Having established the values for statistical significance (P), effect size and power, there exists an assortment of tables and algorithms for calculating the required sample size. These will provide an approximate guide, so if, for example, a required sample size is 284, it is in practice rounded up to 300, in order to account for data loss. Computations (e.g. resource equations or distribution functions) can be cumbersome, but there exists a number of online solutions, equations embedded in statistics software, or tables for calculating the sample size needed for a given level of power – or vice versa (see e.g. Cohen 1992). Based on Cohen's tables, for example, in order to detect a large effect size ($r=0.5$), 28 players are needed, vs. 783 for a small effect size ($r=0.1$).

Let's take an example to show the concrete sampling decisions and concerns in more detail. Let's say we want to investigate whether players in November spent a significantly different amount of money on in-game purchases than the players in October. The null hypothesis states that the mean scores of money spent is the same between the two groups. We employ a t -test to evaluate whether the mean scores differ significantly (see Pearson 1900). Using standard parameters, we say we want to be able to detect a small effect size ($r=0.1$), have a power of 80% (Cohen's $d=0.8$) and a significance level of $p=0.05$. Using Cohen's power tables for effect size (found online, for example here: http://psych.unl.edu/hoffman/Sheets/Workshops/Power_Tables.pdf), we find that we need a sample of 1,571 players. In order to be 99% sure to detect a small effect size, we would need 3,675 players – something that is probably only possible when dealing with either a big beta test for an MMORPG or post-launch data. However, if dealing with a larger effect size, e.g. $r=0.5$, we would only need 64 participants – a much more realistic number in a during-development playtest. Note that these numbers are for simple one-variable situations. When multiple variables are involved (and these interact), the sample size calculations can become more complex.

To quickly summarize the main points:

- Sample size determination depends on the desired P value (the statistical significance or probability value) the statistical power of the test being used and the effect size we attempt to detect.
- Any analysis using sampling should have a high enough power to achieve the stated aim, and calculating necessary sample sizes should be done when the analysis or query is designed.
- Estimating effect size can be difficult and often will rely on existing evidence and previous experiences; conversely, given a dataset situation, a strategy could be to use samples large enough to detect small effects.

9.2.2 Validity

Related to the size of the sample is the internal and external validity of the sample. Internal validity between samples is a feature of the variation between samples.

External validity is a measure of how well the samples properties reflect those of the population.

9.2.2.1 Internal Validity

Internal validity reflects a major goal of sampling, that of consistency between multiple samples. If, for example, completion time of level 2 of Tomb Raider: Underworld is sampled every day over a month, it is essential that each sample is drawn in the same way and using the same size, in order for the results of the two samples analysis to be comparable. If not, there is a risk of introducing biases and/or inconsistencies in the results, which can result in design changes that make a game less enjoyable. Random sampling – where each data point has an equal chance of getting selected – is the most intuitive way to avoid this kind of sampling bias. However, care must be taken to ensure that the sampling process is actually random.

The simplest way to evaluate internal validity between samples is to calculate the mean value of each sample, and subsequently calculate the standard deviation of the sample means. The bigger the standard deviations the more variance between the samples is. Note that the variance may occur due to problems with internal validity or changes in the underlying feature being measured.

9.2.2.2 External Validity

External validity is dependent on the sampling method and the size of the sample. The single largest goal of sampling is to have a sample that can, with a reasonable amount of certainty, represent the overall population or dataset. That is, we want to make sure that the player behavior we observe occurring in the sample is representative of the entire population of players under scrutiny.

Drawing wrong conclusions based on samples can lead to disastrous effects – consider, for example, an analyst concluding that an instanced dungeon in an MMORPG is much too hard, based on a biased sample. Based on his recommendations, the difficulty of the dungeon is drastically lowered, leading to players suddenly gaining easy access to powerful items, ruining the PvP-balance of the game, causing uproar in the player community (providing a real-life example would be too embarrassing for the authors).

Checking whether a sample is representative of the dataset or not can be done by either drawing a lot of samples (ideally hundreds) and checking whether the standard deviation of the mean values of these samples is high. If it is, it tells us that a given sample may not be representative of the population. In practice, it is however easier to estimate the standard deviation of the population, which is called the standard error or error rate. The algorithm for calculating standard error can be found in any introductory statistics textbook (e.g. Field and Hole 2003; Field 2005), but basically the standard deviation of a sample is divided with the square root of the sample size. This means that the error rate for a given sample size scales by the square root of the population.

9.2.3 Outliers

As mentioned previously, outliers in data refer to the data points on the extremes. For example, a dataset might show that 99% of players complete a specific mission in 20–23 min, but 1% take less than half of that time. Outliers could also represent players that are having a lot of difficulty playing the game or cheating to level up more quickly or never die in a multiplayer match, ruining the experience for other players.

Outliers are, perhaps not surprisingly, common in game telemetry datasets. We usually find players who find enjoyment in playing games in ways that deviate from the “norm”, and some games do not really have any standard ways of being played but rather offer a sandbox of environments and mechanics, e.g. MMO’s. Outliers can, therefore, be highly important to consider when working with game telemetry.

Outliers can be identified while working in the parent dataset or during pre-processing of one or more samples. Outliers can be caused by bugs in the telemetry collection code and the associated hardware, or represent actual abnormal player or server behavior. Irrespective of the cause, outliers can be viewed as either noise or something of interest depending on the analyst’s focus. For example, abnormal or missing data points can be used to find bugs in the telemetry collection and storage application. Conversely, if outliers represent actual player behavior, they have historically proven to be interesting in game analytics. For example, they may represent players who are exceptionally innovative, who play the game in a surprising way, who have a hard time getting through the game, or support enormous social networks. Conversely, outliers can represent bots, gold farmers or cheaters (see Chap. 17). In the case of F2P-games, inspecting outliers can also provide insights into why players leave a game (and thus lost revenue can be recovered). If the focus of the analysis is to understand how the game behaves in the extremes then outliers can be important as they offer valuable insight into possible compulsion chains within the game itself.

When drawing a sample of a population the probability of a specific outlier being included is relatively small. For example, if the population is 100,000 then a single outlier has a $\sim 1/100$ chance of being included in a sample of 1,000, but outliers may skew the sample heavily if included – especially for small samples. Therefore, outliers are not always useful to work with. For example, if we are interested in average behavior patterns, or what the majority of players are doing (such as “how long does it take players to get through level X?”), or more sophisticated data mining operations, e.g. machine learning, outliers are often removed from a sample because they risk adding noise and skewing results. Take the same *Tomb Raider: Underworld* dataset used above when discussing sample sizes (roughly 200,000 players). In a 100 player sample, showing level completion times between 20 and 30 min for level 2, the inclusion of an outlier with a completion time of 41 h (2,460 min – there actually was a player of the game who spent 41 h completing level 2 – possibly due to an Xbox left running with the game activated), will skew the results appreciatively, with

over 20 min of playtime on average. When utilizing machine learning techniques, outliers can cause the models developed based on these techniques to be overly complex or lower the external validity (see Chap. 12 for more on game data mining).

9.2.4 Independence of Sampling Units

Sample units need to be independent – if not, probability analysis cannot be performed using standard approaches. If, for example, the behavioral metrics from 1,000 players from 5 different puzzle games were sampled, and we related their completion rates with the relative difficulty of each game’s puzzles, it would be necessary to take into consideration that there are only 5 games, not 1,000. Therefore, analyzing the data at the level of the player will give incorrect estimates of the relationship between completion rates and puzzle difficulty. This is an example of a situation where cluster sampling is useful in game telemetry analysis.

9.2.5 Missing Data

Even with the best telemetry system, data are sometimes missing. Missing data can be due to network problems, issues with the client, the system code or alien invaders. There are two main types of missing data: (1) a player/participant is missing and (2) data from a player/participant is missing. The problem with missing data is that what has not been recorded it is not often random, which can bias analysis results. It is, therefore, important to be on the alert if data are missing. Missing data can sometimes be filled-in (imputed) – which enables the full sample size to be used in model generation – or the data reweighted, which causes the sampled data to more closely match the parent dataset, but both procedures can be complex and time consuming (see Han et al. 2005).

9.3 Sampling in Game Development: Factors for Consideration

Before discussing the different sampling methods and techniques, it is important to consider the context in which sampling takes place for games. In a game development environment, there are three factors that guide the data collection process. The first is the timing of the data collection within the production process: pre-launch or post-launch. The state of the game within the production cycle and the questions imposed by the design team will impact the data collected. In addition, the population of players pre and post launch is also very different. The word ‘population’ is used here to denote the total number of players of a game, which can easily be several million for a popular title after launch. Pre-launch game developers usually constrain

the number of players they invite to see their games, and thus the number of players is different. The second factor is the temporal dimension. The evolving nature of a game and the changes in its population (players) and their behavior after launch (especially true for MMOs) requires the consideration of time-related effects, as compared to an analysis that represents a snapshot of a given population. The third and last factor is the stakeholders involved. As discussed in Chap. 3, there are different stakeholders that are interested in the data; who the stakeholders are usually impacts what data is gathered or what subset of the data is of interest. Below we discuss these factors in more detail.

9.3.1 Data Collection Before and After Game Launch

From a top-down perspective, there are two different situations where sampling is important to Game User Research, depending on the current state of the game, specifically: pre or post-launch. These two situations exemplify the basic two ways of sampling in the present data-driven environment: population sampling and database sampling.

9.3.1.1 Population Sampling

Population sampling (statistical sampling) refers to the process of selecting a group of representative individuals from a population, for the purpose of statistical analysis. During pre-launch, data is usually collected from playtesters (recruited through advertisements within the community), colleagues, and co-workers. Typically, there are relatively few individuals in a sample, ranging from 25 players in a playtesting session to few thousand players in an open beta testing. In both these situations telemetry data (or other GUR/marketing data) can be collected. This data is usually collected from a small sample of the total population of players (constrained mostly by the recruitment efforts). The main challenge is to ensure that the small sample recruited is as much as possible representative of the population (e.g. finding the right playtesters given the target market). The sampling techniques used to ensure that the sampled individuals are representative of the population is referred to as *probability sampling*.

Performing population sampling correctly is extremely important to avoid drawing misleading conclusions about the population based on the sample. There are different techniques for ensuring that the data from the sampled individuals (e.g. a group of playtesters) can be used to generate knowledge that is general enough about the larger population (e.g. the total target audience for a game). We discuss some of these techniques in this chapter (Sect. 9.4).

There are several important considerations to take into account when performing population sampling. The most important one is that a game is not finished and often times is unstable; the results of this the analysis of data within this phase usually is concerned with level flow, usability, play experience, and (closer to launch) predictions

of how the gaming population will receive the game (see Chaps. 7, 14, 21, and 22 for more on these topics). This in turn means that during production typically small samples of the total potential population of game players (as few as one) are used in any single playtest (or study).

9.3.1.2 Participant Selection

Care must be taken when sampling participants to consider how the participants were contacted (were they randomly called? E-mailed?) and any connections between them (did you ask one participant to bring friends?). The major concern with game user research during production is not data overload, but rather making sure the participants accurately reflect the population. A problem discussed above.

Considerations about age, preferred genre, and game expertise need to be taken into account when considering participants; any systematic bias in the selection process could result in data that does not accurately reflect the population, and thus, would result in recommendations that may not improve the game. There are many potential sources of bias (e.g. people who are fans of your games will be more positively biased in playtests than people who are not, etc. This subject is complex, we only scratched the surface here, thus we would recommend (Field and Hole 2003; Kuniavsky 2003) for a good basic introduction on recruiting for experiments) and (Fullerton 2008 and Lewis-Ewans 2012) for information specifically oriented at playtesting.

In summary, to be able to draw any statistically meaningful answers from a test or analysis, at least 20–30 participants are needed. However, the exact number is a debated subject in statistics, and also varies depending on the statistical test/data mining algorithm in question (Han et al. 2005), and the makeup of participants should reflect the composition of the population as much as possible. For example, if a game is aimed at the 18–35 year male segment living in Canada, the people we ask to playtest a game should not be 16–18 year old females from India.

9.3.1.3 Dataset Sampling

Following launch, telemetry data can be collected from every single player in the population, if so desired (see Chaps. 6 and 7). In such a case, working with the entire database may not be possible due to the sheer amount of recorded data. Thus, sampling becomes the only solution. Working with samples provides quick estimation of the issues that may arise with choice of algorithm, settings, data quality etc. Once we are satisfied with a model based on a sample, a larger sample can be chosen to see if model quality improves. If not, there is no reason to spend resources on increasing the sample size. The main challenge in this case then is to ensure that the samples we select are representative of the distributions of the entire dataset. In other words, the challenge in database sampling is to ensure that the samples chosen do not reduce the quality of the resulting knowledge more

than is acceptable, compared to analysis of the entire dataset. Database sampling as a discipline rests mainly in the domains of data mining and machine learning, see Chap. 12 for more information on game data mining.

After game release, the fundamental challenges with sampling for behavioral analysis of the player population are somewhat different from the ones experienced during production as described above. Rather than worrying about getting participants that reflect the population, telemetry can be gathered from the entire playing population that has an internet connection (at least intermittently for the transfer of telemetry data). Thus, we know that the players we collect data from are actually members of the population of players, although this population may of course be internally varied which may require the use of sampling techniques that account for this (see Sect. 9.4). Our concerns at this stage are centered on: how to sample the incoming torrent of data and what variables (behavioral features) to collect for each participant. Other challenges occur within this stage (which we have experienced first-hand). These challenges are centered on data overflow, problems with bandwidth usage and storage and processing constraints.

There are two fundamental strategies that can be adopted when choosing to collect behavioral telemetry: *Deep telemetry*, collecting data on all or a substantial fraction of the possible player behaviors/variables. This strategy permits a broad range of questions to be answered. *Shallow telemetry*, collecting data on only a few behavioral variables. This strategy will only allow for a few, pre-defined questions to be answered but requires less bandwidth, processing and storage capacity. These two strategies can be combined with a sampling approach in different ways to handle the sample representation issue described above:

- **Partial deep telemetry:** Restrict the number of players from whom detailed data are collected, i.e. collect detailed data, but employ a filtering or selection process to choose which clients or servers to collect data from. This sacrifices analytical accuracy (adds uncertainty) as data from only a subset of the population is available;
- **Shallow telemetry:** Collect data from every player but restrict the number of features tracked. This provides analytical accuracy but lessens the explanatory power and ability to drill-down through the data, meaning that some questions from stakeholders might not be answerable.
- **Deep and shallow combination:** Collect detailed data but apply a filter at the collection back end, to ensure that detailed (deep) telemetry is collected for a sample of players, but shallow data is collected for all players. Modifying the parameters of the filter enables throttling of the information stream;
- **Full deep telemetry:** Collect, transform and store deep telemetry from every player, but during analysis extract only the data needed. This solution is intensive in terms of storage and requires careful preprocessing of the data to ensure quick sample extraction and computation at the client side.
- **No sampling:** Collect, transform and store deep telemetry from every player, and perform analysis on the full dataset. If enough resources are available to support this kind an all-inclusive telemetry collection strategy, this is not a problem; however, the reality faced by companies is usually somewhat different.

In summary, the two situations outlined – population sampling and dataset sampling are mathematically similar and have a similar purpose to ensure sample representativeness and coverage, but during the production of a game it is not possible to confirm with 100% certainty whether a sample is representative of the population – we can only estimate representativeness using statistical tests.

9.3.2 *Temporal Properties of Game Telemetry*

Change in both the population of a video game as well as the game itself is another factor that must be recognized when conducting sampling. Video games with heavy multiplayer components are most prone to this problem as they will undergo the largest amount of changes in their lifetime causing some data from earlier points in time to become stale and uninteresting.

In addition, many computer games, particularly single-player games, have a power-log distribution of players with a majority of the players only playing for the first few days or weeks; if actual players are being sampled this means that the players who are still playing 3–12 months after their first session could exhibit drastically different behavior than those playing for 2 days only. In both of these situations, care has to be taken in which player's data to include in the sample as a pure random sample may be systematically biased towards or away from any particular group of players, e.g. hardcore players.

9.3.3 *Stakeholders*

Another aspect to consider in relation to sampling strategies is the involved stakeholders. In any medium to large sized development house, there can be many stakeholders as discussed in Chap. 3. Such stakeholders are: VPs, Marketing, Level/Playfield designers, System/Gameplay designers, Programmers, and so forth. Stakeholders have drastically different needs in terms of what data to collect and how they are to be analyzed and fielded internally. As mentioned by Zoeller (see Chap. 7) (Zoeller 2011), when discussing the *SkyNet* metrics suite used by him at Bioware, a VP is usually only interested in high-level statistics (How stable is the game? How many players are playing today?), Marketing may be more interested in knowing what people enjoy (combat, story, etc.), Designers are more interested in where players go and where they have problems so they can improve the game, and lastly Programmers are interested in system level statistics such as framerate, hardware platforms used and memory usage (see Chap. 3 for further discussion about stakeholders in game analytics, and Chap. 7 for more on *SkyNet*). The varied nature of stakeholder requirements directly encourages a situation where companies try to track as much as possible from every player in the population. A great number of the questions posed by stakeholders, notably at the strategic levels, can be answered via telemetry.

9.4 Sampling Strategies

There are a variety of ways in which a population or dataset can be sampled. Some approaches lend themselves more readily to population sampling or vice-versa. In this section a couple of the most important sampling techniques, for dealing with game telemetry data, are outlined. We also discuss case examples to show how such techniques can be applied. For more on sampling techniques, see e.g. Fowler (2001) for surveys or Han et al. (2005).

The major groups of sampling techniques are **probability** and **non-probability** sampling. The former includes some form of random selection when choosing the data points for the sample, and allows the calculation of statistical chance that the sample is representative of the entire dataset. In non-probability sampling, the data points sampled are chosen non-randomly, and it is therefore less likely that these types of samples are representative of the dataset they are drawn from; however, even while this problem exists, non-probability sampling is useful in specific circumstances.

Different sampling techniques have different pros and cons, and are usually suited for specific situations. An exception may be random sampling, which generally is perceived as the default sampling technique when dealing with quantitative data. However, ensuring randomness can be difficult, especially for multi-modal datasets, and sometimes it makes more sense to tailor the sampling technique to the data. For example, if investigating the behavior of players from four different pre-defined categories (e.g. following a classification analysis); or when performing A/B-testing, stratified sampling is the optimal approach to ensure each category is equally represented irrespective of the percentage distribution of the categories in the parent population.

Sampling techniques applied to game telemetry data is, as mentioned in the introduction, something that is rarely reported on. Even within academic research, very few of the published works discuss how samples of telemetry data were obtained or the criteria imposed on sampling (Chittaro et al. 2006; Gudmundsson 2009; Kim et al. 2008; Thawonmas and Iizuka 2008). However, some publications have included details of the sampling techniques used (e.g. Drachen et al. 2009; Gagné et al. 2011; Drachen and Canossa 2011).

To demonstrate the various sampling techniques, a generated dataset is used. The script for generating and sampling the data was written in Python 3.1, graphs made in R (a highly flexible tool for data analysis). The dataset describes a population of one million data points generated following the log-normal distribution, with a mean of 5 and sigma of 0.7 (see Fig. 9.2). The choice of the log-normal distribution was made, because it is commonly found in game telemetry data dealing with players and time. For example, how long it takes players to get through a particular level. There is a time at which most players finish the level (the median), a minimum time it takes to finish the level (how long does it take to walk or crawl it), and lastly there is a long tail in the distribution of people who got very lost, left the console/PC on while away over the weekend, or some other reason entirely (e.g. buggy telemetry logging code).

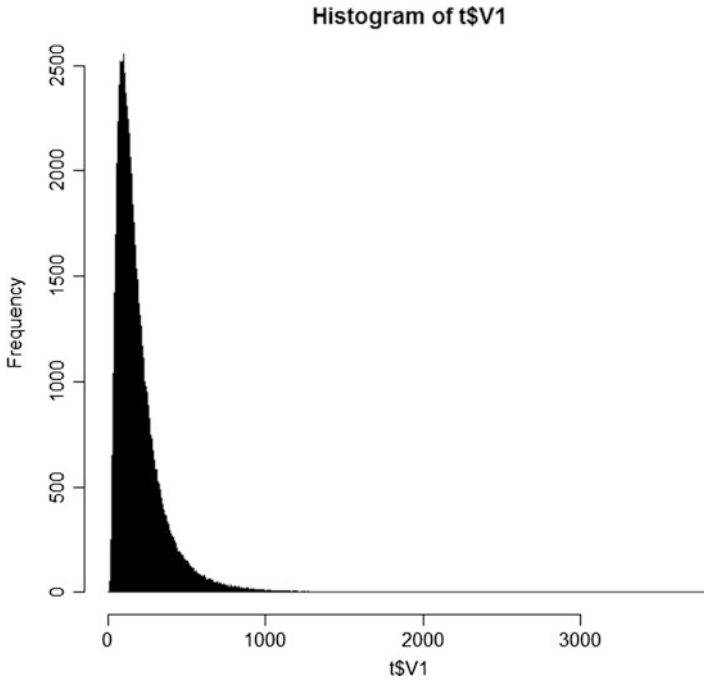


Fig. 9.2 A generated log-normal distribution of the game completion time of one million players from a fictive game

9.4.1 *Random Sampling*

Random sampling is the simplest and most widely used technique for sampling game telemetry data (that we know of), but also one that is dangerous to use uncritically, especially when there are structures inherent in the data. Even a truly random sample does not guarantee that all values represented will occur in the sample, and runs the risk of missing entire small clusters (Cochran 1977). Additionally, determining what random sampling is in specific contexts is challenging.

Using the random sampling strategy, a number of items (or data points) are randomly sampled from the population or dataset (the two terms are used interchangeably here). If a truly random generator is used to determine if an item is included, then the resulting sample should reflect the population to a strong degree.

There are different forms of random sampling that subtly apply bias to the results. For example, all of the items could be numbered and then items randomly chosen, either allowing **duplicates** or not, or each item could be included with a **set – or fixed – probability**. Allowing duplicate items into a sample may be desired if a truly random sample is required but they will, obviously, add more weight to any statistical values calculated based on the sample if they by chance are selected more than once (this can be problematic if outliers are selected multiple times).

Not allowing duplicate items in a random sample requires a ‘bag picking’ randomness in which every item is ensured uniqueness, but there could be a very subtle bias as a result of throwing away results from a random number generator (which includes duplicate values). Given a large enough population, these two sampling methods will provide similar results, however, if we sample more than 10% of the population (often happens with analysis of telemetry data of data from user testing where the population can be limited), the difference between the conclusions obtained from the two forms of random sampling may be significant. Applying a fixed probability to every item in a population to determine its inclusion in a sample means that the sample will not have a fixed size between multiple runs (even random number generators have some variance (Cochran 1977)) and different populations (which is important in post-release telemetry gathering as the population is constantly growing – so successive samples drawn using fixed probability will be progressively larger).

In our experience, questions relating to the entire dataset or population are best sampled using simple random sampling. Questions such as ‘How many players played last week/month?’ or ‘What percentage of players starting this mission completes it?’ are good examples. When dealing with continuously incoming or time-dependent data, e.g. from an MMO beta test, samples should be discarded and redrawn in order to make sure the data being worked with represent the newest build of the game.

9.4.2 Systematic Sampling

Systematic sampling is a variation on random sampling where instead of a purely random set of items (data points) being chosen for the sample, the items chosen are instead chosen on an interval from the initial population (N). The approach runs as follows: determine the desired sample size, and then list all of the items of the population. Then determine a random starting point and then choose every i th item in N , where the interval $i = (\text{population/sample size})$ (Sudman 1976). For example, if a population consisted of 10,000 items but the sample size is 1,500 the sampling interval (i) would be: $10,000/1,500 = 6.67$, which could be rounded up to 7. With for example a random starting location at 4,379; we would choose the 4,379th item of the population and then every 7th from that point, until 1,500 points had been selected (Fig. 9.3).

Systematic sampling works in either the pre- or post- production phases of video game development; either the database of telemetry or the list of potential participants could be sampled in this manner. Systematic sampling is easier to use than a purely random sample when working with physically printed lists. The major weakness to systematic sampling is if there is some inherent pattern in the way the population is listed (i.e., if even items = male and odd = female (Sudman 1976)), this will bias results. However, if no such pattern exists then the sample can generally be treated as being random. Given this weakness and the digitization of nearly every list of a population or dataset in game development, purely random samples are more common.

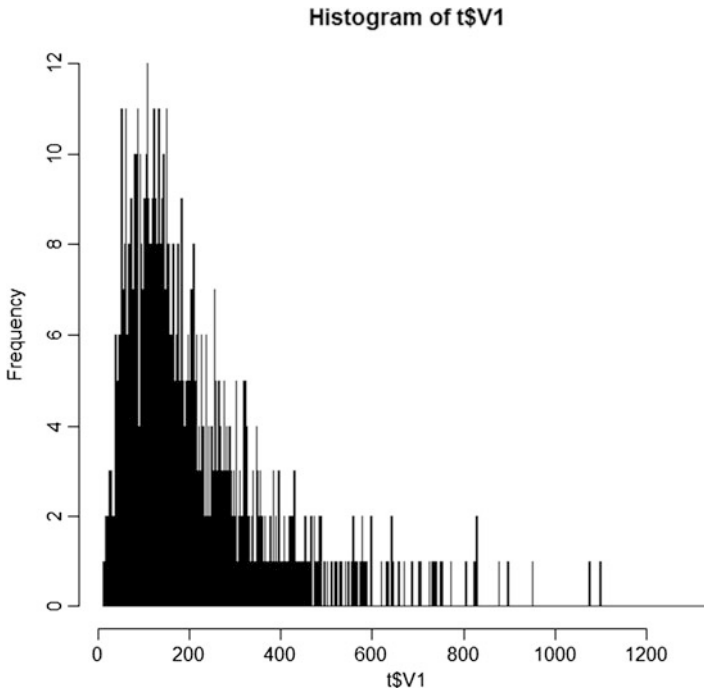


Fig. 9.3 A systematically chosen sample taking every 667th item (data set size/sample size=666.66), visualized with 1,000 bins. The distribution is similar to the random sample of 1,500 items in Fig. 9.1

9.4.3 Cluster and Stratified Sampling

The above sampling strategies assumes that each player being sampled is part of a single population. Sometimes there are natural groups or patterns in telemetry data, however, and in these situations adopting sampling strategies that account for these patterns is useful.

9.4.3.1 Cluster Sampling

This is a technique by which the population or dataset is broken into mutually exclusive clusters; sampling is then performed from each cluster and then reassembled into one sample hopefully representing the variance between and within the clusters. For example, keeping telemetry data of a subset of players (but keeping all of it) is a form of cluster sampling. If any filtering of telemetry is occurring then clustering is probably already being used; for example, if a random amount of multiplayer matches are being recorded then clustering of events such as players who played a particular map or death events for the map are already being sampled using clustering.

Cluster sampling can be used in both pre and post production analysis. For example, pre-production player studies can try to take all potential participants from various geographic locations or contexts (e.g. guilds) but not others, this would allow for comparison across seemingly homogenous clusters. Cluster sampling based upon player ID is a common post-production sampling technique.

9.4.3.2 Stratification Sampling

This is a sampling technique that breaks the population into subgroups based upon specific variables of interest (such as age, geographic location, etc.) and then an equal proportion of samples from each group are randomly sampled. The strength of stratification sampling over a pure random sample is when trying to look at uncommon behaviors, (i.e. matches for a little played map) a representative sample for that behavior is sampled. How to break up (stratify) the population is the hardest question when dealing with stratification sampling as it requires the understanding of the population or dataset, and as a result data analysis is usually iterative with (more) stratification sampling coming in after some level of analysis has been conducted.

Stratification sampling can be used in both pre- and post-production of video games. In pre-production potential participants can be stratified by variables of interest (such as gender, game experience, age) before inviting them in to participate in a playtest.

9.4.4 *Non-probability Sampling Methods*

There are several non-probability sampling methods which are, however, of limited use because they do not allow the calculation of confidence. However, they can be useful when performing population sampling and are therefore briefly introduced here.

9.4.4.1 Quota Sampling

This is a technique that sees a population first segmented into mutually exclusive sub-groups, similar to stratification sampling. Subjective judgment is then used to select the people (or data) from each segment based on a pre-specified proportion. For example, it may be wished to sample 300 female and 600 male players. This allows for targeted sampling, but makes the sample non-random and possibly biased.

9.4.4.2 Snowball Sampling

This is an interesting sampling technique that can be used in situations where the resources available for user research are limited, or when trying to recruit participants from a difficult-to-reach population (Goodman 1961). Snowball sampling is

carried out by initially recruiting one or more members of the target audience, and asking them to name or recruit K more participants/members of the target audience. Once the desired threshold has been reached, sampling stops. This type of sampling is relevant when it comes to finding people to test a game and thus deliver telemetry data, at a very low recruitment cost. Snowball sampling is notably useful when the population that a studio is targeting is not familiar to them, for example a studio of mostly 20–30 year old males making a social game for 40+ housewives.

Quota and snowball sampling are not relevant when it comes to sampling from a database post-production, because at this point in the development cycle, the population is known and recorded. However, the methods are useful when resources are limited and accuracy is not important.

9.5 Best Practices

Rounding up, the following points summarize the key takeaways from this chapter, and comprise some of the “best practices” that we utilize for conducting sampling on game telemetry data:

- **Understand sample sizes.** Calculating the sample size needed in order to obtain a result of a desired significance, with a desired statistical power, is important to ensure reliable results. Understanding Type I and II errors, effect sizes, power and the relationship between samples and populations is the first step for people interested in working with samples.
- **Real world data can be noisy and incomplete.** Data from the real world can be very noisy and it is therefore important to make sure that sources of noise or incompleteness (such as players disconnecting part way through a game, gaps in logging, and similar sources of noise, gaps or bias) are understood, controlled and/or accounted for (see e.g. Drachen et al. 2009).
- **Check for normal distribution.** Human attributes (such as height) are normally distributed, the results of their behaviors when playing games (such as how long they took on a level) may not be. Thus, it is important to always check whether sampled data are normally distributed before running any tests or algorithms that require data to be normally distributed (e.g. a parametric test).
- **Check sample vs. dataset means.** If sampling from a database, where the population mean can be calculated, always check sample means against dataset means – if substantially different, re-sample – by sheer random chance we have sampled from only one section of the dataset. Alternatively, we may have sampled from the wrong dataset (human error happens).
- **Check for outliers.** The purpose of the analysis and the stakeholders within the company will impact whether the analysis is focused on outliers (which provide interesting stories, ideas and innovations; and are used to identify cheaters, bots etc.) or the major distribution (which provides the average actions of the player base) (Thawonmas and Iizuka 2008).

It should be noted that recent evidence from both industry and research, indicates that at least some player behaviors, for example playtime (Bauckhage et al. 2012) and purchasing behavior in Free-to-Play (F2P) games (Lim 2012) follow a power law distribution. This adds emphasis to the importance of outliers, as under a power law distribution, there will, for example, be a small amount of players who spend a lot of money purchasing items or other advantages in a F2P, to the degree where the outliers account for a large percentage of the total sales. It is getting common to hear analysts talk about “winnows”, “dolphins” and “whales”, the latter describing a minority of the players who are, however, highly valuable due to their spending patterns (see also Chap. 4).

- **Populations and design change.** The population of a computer game may change over time as highly dedicated fans finish the game and new ones join in. It is important to monitor the populations from time to time and determine if any new measures need to be collected or samples (and clusters, stratifications etc.) need to be redone, notably in persistent games. Changes to a game’s design will change player behavior and therefore imposes the same requirement for re-sampling.
- **Use iterative sampling strategies.** When initiating a new analysis series, it is usually a good idea to be iterative in the sampling process, and analyzing larger samples (or the population at a shallow level) may lead to an understanding of important clusters or patterns in the data, that can then be sampled from.
- **Automate sampling, analysis and reporting.** Telemetry analysis takes time, and the more they can be automated the better, freeing up analysts to engage in exploratory analysis or simply be able to answer more questions from the various stakeholders. Telemetry analyses that will be run regularly during a production or post-launch, for example playtime monitoring, are ideal targets for automation, and allow for easy comparison between changes in a game (for example around a patch update) (Kim et al. 2008; Drachen and Canossa 2011).
- **Listen to instinct.** Sampling, statistical analysis and data mining has a strong human element, and, therefore, errors can occur (Han and Kamber 2006). Therefore, if a designer or other stakeholder gets a mental alert signal from a quantitative analysis, it is worth checking the result before implementing design changes. Human error happens, especially in a high-stress/limited resource environment like game development.
- **Behavioral telemetry vs. attitudinal data.** On a final note, it is important here to reiterate that telemetry collected from a video game does not include attitudinal data. From telemetry data the reasons for observed behaviors can sometimes be inferred, but not proven – to do this, telemetry data needs to be triangulated with other data, e.g. from surveys (see Chaps. 21, 22 and 23; or see: Pagulayan et al. 2003; Kuniavsky 2003; Kim et al. 2008; Isbister and Schaffer 2008; Drachen and Canossa 2011 or Lewis-Ewans 2012). Relying solely on telemetry makes it difficult to interpret the motivations for the behavior of the players, which can notably be problematic in terms of providing feedback to designers (see Chaps. 14, 21, and 22). For example, observing telemetry data that suggest a particular

way of solving a quest is very common in the player base might lead to the conclusion that the quest provides an enjoyable experience – but this is not a valid conclusion to make, only to infer. There can be other reasons why the quest is frequently selected – e.g. precursor for an enjoyable quest which the players know about.

9.6 Conclusion and Next Steps

In this chapter, we provided a brief introduction to sampling in the context of game user research, with an emphasis on behavioral telemetry, and with consideration as to the context sampling takes place in, and the practical process. The basic statistical considerations, such as determining sample size, have been outlined, and various best practice advice provided. However, sampling is a big topic and certainly not covered 100% in this chapter. Readers new to analytics and sampling are encouraged to investigate statistics textbooks like (Field 2005), data mining textbooks like (Han and Kamber 2006) or for survey work books, like (Fowler 2001; Lohr 2009), before making important design decisions based on sampled data (or telemetry data analysis in general).

It can be a daunting task, at times, to determine what to sample and how to sample it, but the rewards from game analytics can be great. Sampling saves resources in game telemetry analysis, and furthermore allows for rapid iteration of analyses. Correct sampling of the relevant population can give fairly accurate to extremely accurate (depending on the methodology used) reflections of the overall datasets or populations while taking a comparatively small amount of space and resources to analyze. There are several types of sampling that can be conducted depending on the amount of analysis that has already been conducted and the point in development the game is in.

These are the fundamental incentives for adopting a sampling strategy; however, sampling requires a minimum of statistical knowledge and some information about the structure of the data being worked with. In this chapter we have hopefully provided a starting point for obtaining the insights necessary.

About the Authors

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user

research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

André Gagné is a game user researcher at THQ. He received his Bachelor's degree from UBC, Computer Science and his master's from Simon Fraser University. His Master's work investigates the analysis of player progression.

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. Magy worked collaboratively with *Electronic Arts, Bardel Entertainment, and Pixel Ante*.

References

- Bartlett, J. E., et al. (2001, Spring). Organizational research: Determining appropriate sample size in survey research. *Information Technology, Learning, and Performance Journal*, 19(1), 43–50.
- Bauchhage, C., Kerstin, C., Sifa, R., Thureau, C., Drachen, A., & Canossa, A. (2012). How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *Proceedings of IEEE computational intelligence in games*, Granada, Spain.
- Chittaro, L., Ranon, R., & Ieronutti, L. (2006). VU-Flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(6), 1475–1485.
- Cochran, W. G. (1977). *Sampling techniques*. 3d edn. New York: Wiley.
- Cohen, J. (1992). A power primer. *Psychological Bulletin*, 112(1), 155–159.
- DeRosa, P. (2007). Tracking player feedback to improve game design. *Gamasutra*, 7th of August 2007, available from: http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_php
- Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behavior in virtual environments. *International Journal of Arts and Technology*, 4(3), 294–314.
- Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the 5th international conference on Computational Intelligence and Games* (pp. 1–8). Piscataway: IEEE Press.
- Field, A. (2005). *Discovering statistics using SPSS*. London: Sage Publications Ltd.
- Field, A., & Hole, G. (2003). *How to design and report experiments*. London: Sage Publications.
- Fowler, F. J. (2001). *Survey research methods*. Sage Publishers.
- Fullerton, T. (2008). *Game design workshop: A playcentric approach to creating innovative games*. Amsterdam: Morgan Kaufman.

- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2011). A deeper look at the use of telemetry for analysis of player behavior in RTS games. In *International Conference on Entertainment Computing (ICEC 2011)*, Vancouver, Canada.
- Goodman, L. A. (1961). Snowball sampling. *Annals of Mathematical Statistics*, 32(1), 148–170.
- Gudmundsson, E. (2009). EVE Online | EVE Insider | Dev Blog. <http://www.eveonline.com/devblog.asp?a=blog&bid=686>
- Han J., & Kamber, M. (2006). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann Publishers.
- Han, J., Kamber, M., & Pei, J. (2005). *Data mining: Concepts and techniques* (2nd ed.). San Francisco: Morgan Kaufmann.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. New York: Morgan Kaufman.
- Kennerly, D. (2003). Better game design through data mining. *Gamasutra*, 15th August 2003, available from: http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_php
- Kim, J. H., Gunn, D. V., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008) Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In: *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI'08, Florence, Italy.
- King, D., & Chen, S. (2009). *Metrics for social games*. Social Games Summit, San Francisco.
- Kuniavsky, M. (2003). *Observing the user experience: A practitioner's guide to user research*. San Francisco: Morgan Kaufman.
- Lewis-Ewans, B. (2012). Finding out what they think: A rough primer to user research, parts 1 and 2. *Gamasutra*, 24th April 2012. URL: http://www.gamasutra.com/view/feature/169069/finding_out_what_they_think_a_php
- Lim, N. (2012). Freemium games are not normal. *Gamasutra*, 26th of June 2012. URL: http://www.gamasutra.com/blogs/NickLim/20120626/173051/Freemium_games_are_not_normal.php
- Lohr, S. L. (2009). *Sampling: Design and analysis* (2nd ed.). Boston: Brooks/Cole.
- Pagulayan, R. J., Keeker, K., Wixon, D., Romero, R. L., & Fuller, T. (2003). User-centered design in games. In *The HCI handbook*. Mahwah: Lawrence Erlbaum Associates.
- Pearson, K. (1900). On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine Series 5*, 50(302), 157–175.
- Sudman, S. (1976). *Applied sampling*. New York: Academic.
- Thawonmas, R., & Iizuka, K. (2008). Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology*, 2008, 1–9.
- Zoeller, G. (2011). *Game development telemetry*. Presentation at the Game Developers Conference 2011, San Francisco, CA.

Chapter 10

WebTics: A Web Based Telemetry and Metrics System for Small and Medium Games

Simon McCallum and Jayson Mackie

Take Away Points:

1. Description and documentation of an open-source C++/HTML-based tool for logging, storing and analyzing game telemetry for small-medium sized games.
2. Discussion of some of the key design concerns that emerged during the development of the drop-in style tool for game telemetry and metrics.

10.1 Introduction

In the 1980s game developers were able to pride themselves on developing every part of a game, sometimes even to the hardware being used. However, with the increasing size of game development teams, and the increasing demands on smaller game titles, the use of middleware and other third-party tools are becoming increasingly important to the industry. While there are still game developers who are reluctant to employ code not developed in-house, this position is getting harder to justify with the changes in the marketplace and development strategies. An example of this can be seen with the widespread use of game engines from third parties, e.g. the Unreal Engine and Unity 3D.

Metrics are an ideal candidate for a middleware tool, as discussed e.g. in the Unity interview in Chap. 8, and in Chaps. 6, 7 and 16. Until recently, very few games are sold due to their great use of metrics or the depth of gameplay analysis provided by metrics. However, this is changing with the rise of the Free-to-Play genre, where telemetry analysis forms a means of testing designs that is as important as the design itself (Fields and Cotton 2011). Additionally, game metrics analysis

S. McCallum (✉) • J. Mackie
Gjøvik University College, Gjøvik, Norway
e-mail: simon.mccallum@gmail.com

can be used to make good games better. Thus, the design and development process can also be improved by integrating game telemetry analysis into the incremental development of the game. Large game developers have the financing required to develop and support an in-house metrics system. For example, Microsoft Studios Research have developed TRUE (Kim et al. 2008; Thompson 2007), and other big publishers have similar systems for collecting, storing and analyzing game telemetry and other user research data (Drachen et al. 2009). Indeed with the increase in the use of data driven game design, game data collection and analysis is becoming a larger part of many game companies' budgets (Fields and Cotton 2011).

Small to medium-size developers will however typically, due to financial reasons, need to use middleware or third-party tools rather than invest the time in developing their own solutions. In this chapter, we present the WebTics API – an API packaged as a product that is aimed at smaller developers. We will discuss the design decisions and the approach to the telemetry system that the API enables, in the hopes that developers will be able to simply import this code into their projects and improve the quality of their games through the information it provides.

Middleware tools can be incorporated into games at many levels (Isbister and Schaffer 2008; Zoeller 2011), also see Chaps. 7 and 8. In fact some game creation tools are little more than a collection of middleware, for example Delta3D (<http://www.delta3d.org>). Graphics engines led the way for the acceptance of middleware within the game industry. The reuse of graphics engines and game engines, like Unreal, within game companies led to the realization that the technology developed to create a game could be sold to other companies. The widespread of Unreal and Unity engines by the industry serve as a good example of this. Companies now realize the value of middleware tools.

This expansion of the use of middleware is part of the motivation for our API and tools. In this chapter, we concentrate on *WebTics*, the tool we created, specifically describing the underlying design considerations, as well as three examples of its use. We also include a documentation-style description of the API as an appendix. The chapter is divided into three general components: Sects. 10.1, 10.2, 10.3 and 10.4 is focused on a discussion of the current tools and design considerations of WebTics, Sect. 10.5 describes case studies showing various scenarios of usage (use cases), including programmers looking for bugs, designers balancing a game, and marketers/QA personnel investigating usage numbers. Finally, the Appendix contains a documentation of the WebTics API.

10.2 Types of Metrics Middleware

Game metrics middleware systems can be categorized as: products, services, or as a mixed model. **Products** are systems that you can purchase to add to a game. **Services** are provided based on a centralized-server, or in the cloud, and are categorized as Software as a Service (SaaS). In this model you do not host the software or the data, but merely interact with the system, usually through a web browser. Google Apps

(<http://www.googleapps.com>) are typical examples of SaaS. Mixed systems provide some tools that are independent and some that are hosted by the metrics company.

The business models for each of these types of products are different, as are the best situations to use a particular option. When evaluating the best option for a specific game development situation it is important to assess the advantages and disadvantages of the each type of middleware. Service-based metrics middleware (such as: the ROAR Engine,¹ Kontagent²) include:

- **Automatic updates.** Your metrics tool will always be the latest version, with no additional effort. Note that this could be bad if you are trying to directly compare results and something has changed in the tool you are using to collect data.
- **Fewer compatibility problems.** The web-based link to these services usually runs on any browser; logging calls from your game goes through a standard web interface.
- **No installs.** You do not need a DBA to run a database, network coders to manage user data, or dedicated machines to deal with the potential traffic.
- **Small initial costs.** These systems often ask for monthly fees rather than large upfront investment.

Buying a middleware metrics product, such as Playtomic³ or Orbus Gameworks,⁴ has the following advantages:

- **Control.** Having internal functionality and measurement information stored entirely within the company's intranet may be important to protect new Intellectual Property. Storing this data on a remote server can be a deal breaker for many businesses. Relying on other companies to protect and provide data is a risk.
- **Offline functionality.** In house testing behind a firewall is possible regardless of Internet connection. This may also allow external testers with poor Internet connection to test the game while offline and upload logging events when they reconnect. Only testing games when there is a good Internet connection may leave out specific bugs.
- **Single payment.** Products are purchased once and serve their purpose without usually requiring additional monthly payments. Some companies offer support and updating services for products. This starts to become part of the mixed model of product and service.

WebTics is designed to be a telemetry and metrics product rather than an on-going service, much like Playtomic. The main difference is that WebTics is open source, and provide complete control to the developer using the system. This approach will suit companies that have the internal resources to set up a server with SQL and PHP, and an interest in being in complete control of all their data.

¹ ROAR engine. <http://roarengine.com>

² <http://www.kontagent.com/>

³ <https://playtomic.com/>

⁴ Aleph Metrics Suite from Orbus Gamesworks <http://www.orbusgameworks.com/solutions.php>

We do not provide a side-by-side comparison of the many game metrics systems that are currently available, as a comparison of features would become out of date very quickly.

10.3 Types of Game Metrics

Game metrics can be internal to the game, or recorded by external sources. Internal logging comes from monitoring the actions in the game, either created by the player or from the game itself. There are many types of external sources of telemetry and metrics information. Standard user testing with a trained observer provides large amounts of external information.

The focus of WebTics is to support internal logging, but it is designed to be able to integrate with external systems. The main telemetry included in the systems comes from calls embedded in the game. These internal logging calls are added by the developer during the implementation of the game, usually with a specific logging objective in mind. This type of logging will only be able to tell us about the player actions or the operations on the game client. It does not, for example, give us additional information about the player, such as the context of play, gender, distractions, accessibility issues or how the player experiences the game. External sources of information include biometrics, production metrics and performance metrics. Even within the scope of behavioral telemetry, it is difficult to incorporate different types of sources and types of game metrics data into a single unified resource.

When collecting different streams of data synchronization becomes an issue. There are various ways to engineer the environment of a game telemetry logging system to assist synchronizing different sources borrowing from synchronization methods used by other media. For example, in traditional film production a “clapper” is used to produce both a visual event and an audio event. These events are used to align the audio recording with the video recording. Another example would be using a psycho-physiological device such as electroencephalograph (EEG) which records brain activity, and linking this with a video recording of a player; a simple tap of a finger on a specific sensor will give a detectable event in both data streams (Mandryk 2008). Thus, for synchronization of game telemetry, we can use similar approaches. To assist in the synchronization of various data sources, WebTics provides microsecond timestamps for all logged events. This can be used post-session to synchronize the various sources, including situations where external data are used, e.g. from psycho-physiological sensors (Mandryk 2008).

Once we have created an environment that supports the collection of various metrics sources we can improve the usability of the data by standardizing the way in which we store the logged events. There are a set of features that are common to all events that can be logged. For events to be measurable they must be:

- **Present:** you cannot measure when something does not happen. You cannot measure the fact that there are no bugs in the system. However, you measure when errors occur, and the length of time between errors. It is important to remember to

adage “absence of evidence is not evidence of absence” (Altman and Bland 1995). This focuses the logging on things that can be measured during game play.

- **Observable:** there is no point trying to measure something that cannot be directly observed either as events in the game or by using some external sensors, such as heart rate monitors.
- **Specific:** the event must have a clear definition of when it occurs. If the intention is to measure Actions Per Minute (APM) (Lewis et al. 2011), you must define what is an action. Is each keystroke an action? If do you do for typing in a chat window? Are you interested in all clicks, or only those that relate to changes in game state?

These principles can be used to ensure that the data recorded achieves the desired outcome. Some recording objectives might require additional input external to the game actions, other may need to be described so that what is being recorded has meaning. Understanding the goal of recording data can significantly increase the likelihood of being able to use the data you have collected.

This set of features also help explain why it is difficult to request a programmer to include a metric reading of something like “fun” during game play. Fun would be *present* but it is difficult to identify how to *observe* fun, and even more difficult to describe the *specific* thresholds on sensors that indicate that the player was having fun (Yannakakis 2012).

Events can also be broken down temporally into two different types of events:

- **Instantaneous:** Events that pass a threshold value are registered with a specific time.
- **Duration-based:** Events that span a period of time and are defined by continuing action.

In our event logging system we decided to implement only instantaneous events. This is based on the principle that events that record a duration must have both a *start* and an *end* to define the duration. Therefore, we can store those *start* and *end* events and calculate duration as a feature, with post-processing and data mining. This could be achieved in WebTics by calling (see [Appendix](#)):

```
metricsSystem->LogEvent (EventTypes::DragnDrop, Event-
Subtypes::Start, m.x, m.y, 0);
metricsSystem->LogEvent (EventTypes::DragnDrop, Event-
Subtypes::End, m.x, m.y, 0);
```

The amount of data stored, and the internal logging of the metrics system have a significant impact on the sensitivity and usability of the metrics data (Kim et al. 2008). If you do not record enough detail about the actions of the player, it may not be possible to extract information about play styles, errors, or dominant strategies.

In WebTics we have decided that each event will log, not only the data provided by the programmer, but also three additional data items:

- **Time:** time since logging session started
- **User ID:** associating data with a specific instance and user of the game
- **Game Session:** for associating data with the correct session

This additional data provides the minimum set of information not only for simple aggregate-based metrics (counting events/time period), but also for more complex data mining techniques (see Chap. 12 for more on game data mining).

It is also critical for consistent evaluation of the data that events are associated with the specific **Build Number** that generated the events. In WebTics we do this by linking each Game Session with a Build Number. Build numbers are used in software engineering to specify the exact code used to create an executable. If any part of the code is changed, the build number is incremented. The current version of Microsoft Word used for this document has a build number of 120,421, suggesting that there has been over 120,000 compilations and integration of the application. If a change in the code has resulted in a change in the player interactions, each event must be able to be associated with the specific build. In WebTics the Build Number is added to the OpenMetricSession call at the start of the session:

```
metricsSystem->OpenMetricSession( uniqueID, BuildNumber );
```

With the build number associated with the recorded events we can start to perform consistent data analysis.

For the analysis of data, the initial metrics that can be easily extracted are:

- **Frequency:** how often an event is occurring. This requires that every occurrence of a specific event type be recorded.
- **Correlation:** what events consistently occur with other events. This uses the time and sequence order of the recorded events to find events that co-occur.

Correlation requires more processing and information to calculate than frequency. Correlation can be as simple as determining if event A occurs within a set time of event B. In Diablo3™ on release there was a correlation between giving a Templar follower a shield and having the game crash (Johnston 2012). This is calculated by looking at events in the moments leading up to a crash.

10.4 Design Approaches

Our approach to designing WebTics was a mix of bottom-up and top-down design. Some of the system features originated from analyzing the data that we had previously collected in an adhoc fashion in earlier games, and the others from a top down assessment of what might be needed in future. We combined events that are easy to log with ‘what if sessions’ to generate a wider array of measurements. We believe that this approach to the problem of developing a metrics system allows the developer to both focus on the low-cost high-return data, as well as directing the development toward the ideal metrics. Merely focusing on recording data would not provide room for extensions to complex interactions, while focusing only on the complex metrics would create a large barrier to starting to use the system.

During the design of the system we discussed the types of users and their objectives (Use Case analysis). It is often tempting, as a developer, to start looking at data being generated from your game and get lost in following what looks interesting at

the time. Although this “Grounded Theory” (Charmaz 2003) approach to developing an understanding of the game data as it is collected might work in an academic setting, usually in a commercial environment there isn’t enough time to “muck about” playing with the system without generating actionable recommendations for the game (Pagulayan et al. 2003).

In the design of WebTics we have several constraints. The data must be human readable, encourage and follow good software engineering approaches, and be scientifically and legally sounds. The following subsections describe these constraints.

10.4.1 Human Readable Metrics

One of the potential problems for a number-based metric system, like this one, is ensuring that there is always a consistent translation from a block of numbers into human readable logic. From personal experience, a database of carefully collected metrics becomes meaningless when the link between the meaning of the numbers and the actual numbers is lost. When recording data it is far better from a storage space perspective to store just a numeric value that represents a category, rather than storing a string. Storing numeric values rather than strings requires a translation between the name of an event, for example `PlayerDied`, and the associated number `EventType 3`. This link is sometimes called a Data Dictionary (Dustin 2002). This approach to condense data has implications in terms of introduced biases, as discussed in Chap. 13.

In designing our system it was essential to have a clear way of updating and maintaining the Data Dictionary. Our solution to this problem is to use a three-part system:

1. Include a version and build number when starting the logging process.
2. Include a set of strings that are associated with the numeric event data types.
3. Create a back-end table to associate each event number stored with its name in the current version/build.

This solution allows programmers to use enumerated types for event logging. Enumerated types provide a conversion between the name of a variable and a number, e.g., in Java `java.util.Calendar.TUESDAY = 3`. These enumerations minimize the size of the data stored in the database and gives a unified way of updating the associations between the metrics stored and the events in the real world. The data dictionary allows the translations, so that we can understand what it all means.

10.4.2 Build Numbers

Recording the version and build number for programs is part of good software engineering (Henderson 2008). It takes on additional importance when dealing with metrics data, e.g., for A/B testing features (see Chap. 4) (Fields and Cotton 2011). If the data you collect is not clearly linked to the code that generated it, much of the

value of the data is lost. If you are using a version control system (which should really be standard for any programming project) including the version number of the repository is also valuable for tracking the source that generated the data. Other industries also have a strong focus on version and batch numbers. Medical research, for example, has a very strong requirement that every test is associated with a specific batch of medicine.

The system we have developed does not enforce any particular numbering format for versions, builds or repositories. However, the database back end supports a dot separated format “majorVersion.minorVersion.build.repositoryVersion” (for example “2.1.120.8756”) to perform some additional continuity checks.

10.4.3 Privacy Requirements

One of the significant decisions to be made when logging player’s behaviors is how much information you can and want to log, and to what extent that information can be traced back to an individual. In the US and various European countries privacy and data storage laws provide legal protection to your players. Any metrics middleware system has to be created with the ability to ensure that it complies with the law of as many regions as possible. However, no matter how much care is taken in the development of a telemetry and metrics system, it is still the responsibility of the users of the system, the game developers, to ensure they are operating within the current laws for their country (see references below).

In the design of our system we have included the ability to query and update an authorization table. This table contains the date and level of authorization given by the user. The table includes an expiry date for data related to that user. Rather than automatically deleting the data at that date, the system warns the developer that the data has expired and that the “flush expired data” page should be called. This removes the data from the database.

While this process may seem overly cumbersome for small developers, the legal issues related to storing data can be extremely serious, and small developers often do not have the spare cash to pay for lawyers to defend these cases. We hope to minimize the risks by providing the infrastructure for managing player authorization for data collection. In the default logging of player interaction we suggest using anonymous session IDs rather than unique player IDs unless the developer is using the authorization system.

Regional guidelines:

- For European developers the legislation is from the Directive 95/46/EC of the European Parliament. Search the Europa web site using CELEX and number 31995 L0046.
- For those in the UK, the Data Protection Act 1998 is the relevant legislation, and the Information Commissioners Office is the auditing body.

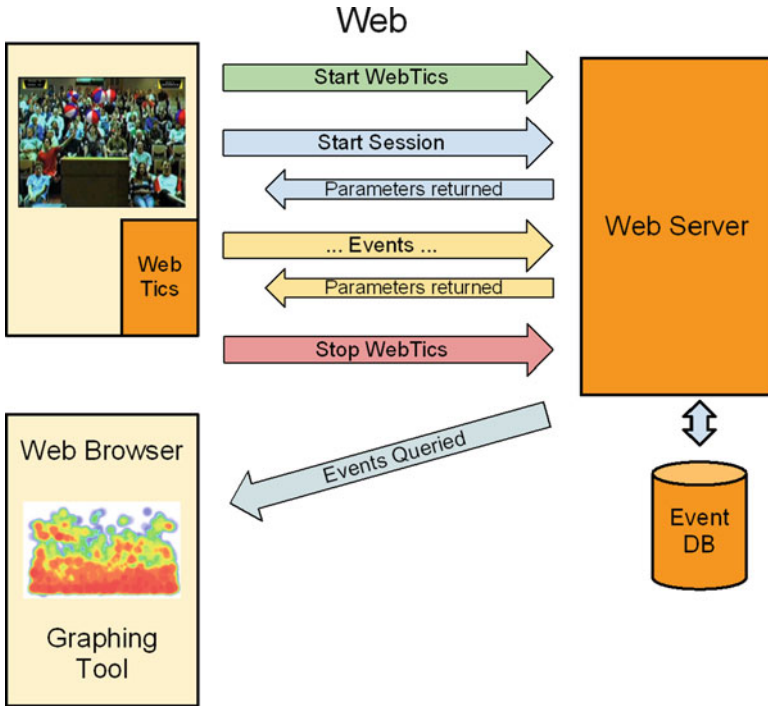


Fig. 10.1 Diagram of the components of WebTics and the interactions between these components

- In the US the FTC Fair Information Practice Principles are the guideline for storing and processing information about users that may be able to identify them.

10.4.4 System Overview

The API for WebTics is shown in the [Appendix](#) at the end of this chapter, However it is useful to have an overview of the system before we move on to discussing various case studies in the next section.

WebTics is designed as a light weight system. Figure 10.1 shows the conceptual modules of the system. The main interactions are handled by simple URL posts to the server that is running the database. These calls contain the event codes, variables, and time of each event, as well as the unique identifier for each session. Normally the API handles the session details, as this allows the programmers to focus on the metrics they are recording rather than the process of storing them. However, there is a clean separation of the database processing and the WebTics API. This allows developers to create their own versions of the API, database, or visualization system.

The workflow for a single game application is:

- Initialize the API singleton, using build number and user id.
- Start a session
- Log events
- Close the session
- Cleanup using dispose on the API logging singleton

Having implemented the logging, and run some game sessions, the database will have a large number of events. These events can be viewed directly as a database table, or using the Javascript based visualization tools.

As part of the set-up for a session the database server can be asked to return parameters. This allows dynamic adjustments to the game without having to recompile and send a new patch. This returning of parameters can also be used to generate interactive questioning systems. When events are sent to the server, there is the option for them to return data. This data could be new parameters, or questions that can be asked of the user.

10.5 Case Studies

It is often beneficial to see how a system might be used before spending the time and resources to test the system in your own environment. For details on implementation and including WebTics in a project see the [Appendix](#) at the end of this Chapter. The following section discusses three examples of the usage of WebTics. If one of the situations is close to the reader's requirements then these cases will hopefully provide a good starting point for considering the use of this metrics' middleware.

10.5.1 Examples of Using Metrics for Designers

The rise of social gaming has created a new model for game development and publishing. Rather than developing a complete product and releasing a full game, social games are usually released early as small games that are then incrementally built as the game becomes popular. This incremental improvement to the game can be informed by players' actions in the current version of the game. This data-driven approach (Rabin 2000) emphasizes the use of data to understand and improve the game, rather than merely use the intuition of the designer (Isbister and Schaffer 2008; Pagulayan et al. 2003; Derosa 2007). There are pros and cons to this approach, as it tends to develop incremental improvement rather than radical new designs. However, to ensure the best possible use of this number-based approach, it is useful to collect large amounts of data from players' interactions with the game. Simple figures such as final score or play duration may not capture the underlying player dynamics essential for improving the game, more information is usually needed.

As designers, we have used data from play-testers in various different ways. When improving a game for stadium audiences called BallBouncer (Sieber et al. 2009), we used the location of interactions to manipulate the “random” generation of balls. This improved the player experience by focusing the interaction on the areas of the stadium where there were active players. The use of player data in the development of our games can be categorized into three different types: game balance, game rule evaluation, game player surveys. These categories will be the subject of the next subsections.

10.5.1.1 Game Balance

In many games it is important that the game is well balanced to allow the player to experience the full range of activity in the game. Complex rule sets make it very difficult to predict all of the possible interactions that will occur during a play session. It is inevitable that some players will think of strategies that none of the design team anticipated. By using a metric system, a designer can identify which players win consistently, and then look at which tactics they are using. These emergent dominant strategies can then be analyzed to decide if and how they should be nerfed (have some aspect downgraded to weaken the overall strategy).

10.5.1.2 Game Rule Evaluation

It is increasingly common in free to play games to perform A/B testing on specific aspects and parameters of a game (Fields and Cotton 2011). The WebTics system incorporates the ability to have feedback from the database to the game. This can be used to update parameters in general, but also given specific user IDs it can serve a different set of parameters for different groups of users. Each user ID can then be associated with a parameter set, and the differences in critical variables, like play time and revenue per user, can be collected and compared. This eliminates the need to have different executables, and distribute them to the different user groups.

10.5.1.3 Game Player Surveys

Finally, the designer can use the in-game question system to survey the players at different stages **during** their game session. The designer can insert events that trigger questions to be sent from the server to the running game. In the game, developers have to insert some way of handling the question’s XML format returned from the server. Given that the questions are tagged to an individual user it is then possible to limit the number of questions asked, or to ensure that the same questions are not asked multiple times.

These in-game surveys results will give designers different information when compared to data from the end of the play-session. There is, however, a risk of getting

poor feedback by harassing the player and breaking the flow of the game with too many questions. This feature should be used in the early testing phase rather than late in production.

10.5.2 Examples of Using Metrics for Developers

Event logs can easily be used for standard bug detection. Looking at the events that are logged immediately prior to an unexpected session end, will lead to finding the event that led to the crash. However the power of a metrics system is the ability to analyze the telemetry data, and find bugs caused by complex interactions. Rather than looking for events that lead to a crash, we are looking for combinations of events that resulted in a bug. The errors may be emergent, only becoming evident after a specific series of game events. By logging each key press or memory event, the recorded metrics may be data-mined (see Chap. 12) to establish if there is a pattern of actions that lead to a crash.

While development is in-house, this information could be logged to disk using standard logging tools. Note, however, that if data is logged in multiple systems, it may be difficult after release to collect all the data logs from the various systems. Using a system like WebTics enables the developer to assess errors on remote systems, e.g. via crash messages.

We have used the system to monitor multi-threaded programs, as these can be particularly problematic. The order of events occurring in a program with several threads often cannot be specifically predicted. The exact sequence of events across different threads may lead to either crashes or thread blocking. Logging of events that lead up to this fail state can greatly assist in tracking the cause of these bugs.

10.5.3 Examples of Using Metrics for Academic Evaluation

An important part of the games industry are games used for purposes other than entertainment, such as Games for Health (Sawyer and Smith 2008), exergaming (Göbel et al. 2010), and Game for Education (Gee 2003). The Wii Fit™, Zumba Fitness™, and BrainAge™ games are commercially successful examples of games that are designed to have effect outside of the game world. These are often collectively called Serious Games. Most of these games are trying to change the player's behavior, knowledge or attitude. To demonstrate the effectiveness of the game in changing the player it is important to provide data supporting any claims, otherwise the intervention would not be evidence based. Examples of these are also discussed in Chaps. 31 and 32 and in the interview with Serious Games's CEO Simon Egenfeldt Nielsen.

Logging of player behavior in a game can be used, not only as a way of improving the game, but also as a diagnostic tool. As part of a Masters project investigating independent play on a tablet quiz game for the elderly (Askedal 2011), it

was important to record the motor performance while playing the game. Clicking performance was logged for both correct answers and all inaccurate touches. The ratio of successful touch actions could then be compared to the number of misses using simple frequency over time. This was used to show the initial learning of the touch-based interface. The success rate in this session creates a baseline for each user. If the current performance on the game interface changes suddenly, the success metric for touch interaction could be used to indicate that there may have been significant changes in the physiological health of the player.

10.6 Visualizing Data

The human brain is amazingly powerful at finding patterns and correlations, so good that it often finds patterns that do not exist. We can, however, use this power to gain insight into the meaning of the telemetry data being collected. Any good metrics API must provide tools for standard visualizations of the data collected. The simplest form of these is the highscore list, which visualizes one dimensional (1D) data set – the metric of how many points the player has scored during the game.

Graphs and charts can be helpful when viewing larger data sets. Graphs are particularly good for spotting trends. A 2D scatter-gram places a different metric on each axis. This allows simple outliers to be identified quickly, and simple correlations between the data to be spotted. In WebTics we use the D³ Data-Driven Documents javascript library (D3.js) (Bostock et al. 2011) for creating visualizations of recorded data. This library focuses on transforming data using web standards.

Given that most games occur in a 2D or 3D environment, and that many 3D games have 2D game play, the (x, y) location of events provides a valuable insight into the data. WebTics includes the time, and critically the location of events. Including the location allows powerful visualizations of the data. When Halo3 (Microsoft Game Studios 2003) was released, the developer, Bungie, provided heatmaps for each level showing spatial metric data, such as: where characters die, where they scored kills from, how far they got when they died, etc. (Thompson 2007). Heatmaps are relatively easy to produce once the data has been collected and organized. A human viewer can quickly identify choke points, sniper locations, and general game imbalance.

Heatmaps can be applied to almost any positional data. The concept is that events create heat at a point and that the event has an area of effect. The larger the number of events the “hotter” the area. The heatmaps in WebTics use a javascript tool developed by Patrick Wied, called *heatmap.js* (<http://www.patrick-wied.at/static/heatmapjs/>). This uses radial gradient functions and a customizable color gradient. The general process for producing a simple heat map image can be achieved by adding a Gaussian distribution to a point (x, y) in a 2D array. The color displayed is selected based on the value in each pixel of the accumulation storage array. The range of the data used to generate the heat map can be adjusted manually to inspect areas where there are few events, and so would have low contrast in the full heat map.

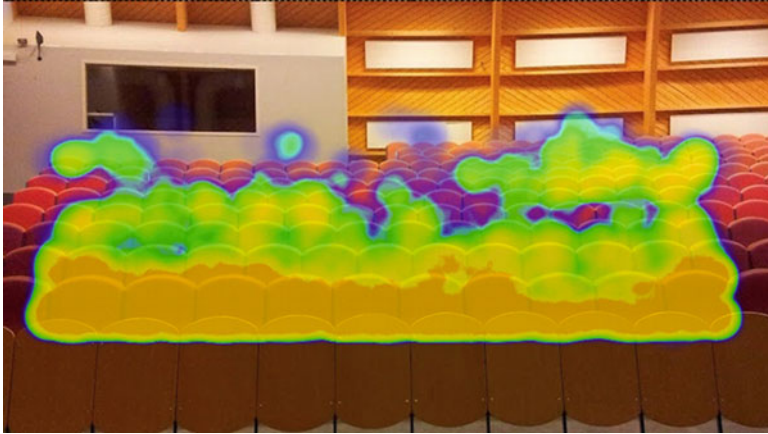


Fig. 10.2 A heatmap produced from the position of ball interactions in the game BallBouncer, seen on the auditorium in which the game was played

Figure 10.2 shows a heat map from the Augmented Reality game BallBouncer (Sieber et al. 2009) used for stadium entertainment. The game logs every successful interaction with the virtual items. From this map it is easy to see that most of the activity is happening near the front of the auditorium. This allows us to adjust the parameters in the game to improve the distribution of interactions.

10.6.1 Data-Mining

The WebTics system provides data visualization tools and very simple data mining tools. At the time of writing, the data mining tools only include frequency and binary correlation analysis. This allows the developer to analyze the frequency of the events that are logged to the system, and find correlations between selected events. This very simple analysis is performed on the server side using PHP. For more complex data mining (see Chap. 12) it is possible to use either the Knime tool initially developed at the University of Konstanz, Germany (Berthold et al. 2008), or for those who are more interested in the low level machine learning and programming analysis the WEKA system (Hall et al. 2009).

10.6.2 Extensions to the API

During in-house testing and academic research it is often useful to have two different computers linked to the same play session. One computer playing the game

while the other computer is recording video and audio from the play session. While it is possible to have two computers with synchronized clocks, the focus of WebTics is to support SMEs (Small to Medium Enterprises), which usually do not need the accuracy of this kind of synchronization. To assist in synchronizing different events the logging system allows the developer to register a pass phrase with the logging system after opening a connection. The secondary computer system sends the logging system the same pass phrase and is given the Session and Game IDs to complete the pairing. The second system zeros its internal timer and then sends all telemetry events with the Session, Game and local time stamp indicating the time since pairing.

The metric server inserts all messages into the database in order, with a time of arrival timestamp. This, along with the IDs and a parameter including local timestamp information, will provide enough data to enable analysis. Machines used in this style of system will not typically be ‘in the wild’, they will be in-house research machines within companies or research labs. In this situation the computers are probably on the same LAN, minimizing network delays. With a dedicated web server to handle all log events promptly, this lightweight, easily established synchronization will be acceptable for most game event analysis.

10.7 Conclusion

In this chapter we have discussed the design of an open source telemetry and metrics system, called WebTics. This system is designed to be lightweight and easily integrated into a game development project. The open source nature of the project will result in significant additions to the tools described here. The description of the design decisions and the principles behind the development of WebTics will hopefully help the reader understand the reason for specific features. The core principle is to provide tools to the developer using simple open formats.

We are currently using WebTics as part of our game development research projects. As such there are plans for several additions to the system, including javascript and linux from ends, tutorials for using the data with Knime and Weka, and potentially an option for the backend database to allow for NoSQL, which would allow for much larger data sets.

There are already a large number of game metrics tools available, but we intend this tool to fill the role of an open source project for small companies to start to use telemetry and metrics. Once the needs are more than what WebTics can provide, it is probably time to pay for professional support, and use a company to provide your metrics tools. For the latest versions of WebTics clone the project from the GitHub repository [git://github.com/SimonMcCallum/WebTics.git](https://github.com/SimonMcCallum/WebTics.git) or fork the project and contribute to the on going development of WebTics.

About the Authors

Dr. Simon McCallum: The background for this middleware tool comes from a combination of academic interest and corporate necessity. Having started researching games for health as an academic at the University of Otago in 2003, it became clear that recording player actions for review and evaluation was essential for any scientific validation of the projects we were conducting. In 2007, I decided to live in the trenches of game development, get a commercial job in Norway. One of the companies I worked for was a game distributor, with an in-house game development team. In this team one of the projects we worked on was the design of a middleware metrics tool to provide to small developers as a service attached to our distribution agreement. This was to be used internally as well as supporting other developers. 2008 was not kind to game companies. Thus, I returned to academia and started teaching game development. The importance of game metrics stayed with me and I used it in student assignments to teach database at undergraduate, and game technology at Masters level. This API is a result of this combination of researcher need, commercial imperative, and educational tool.

Jayson Mackie. With experience over the last 15 years working with Governmental databases, high-uptime telecommunication software and data driven games, I have been in many situations where a simple API to enable customer or player machines to log extensive data to a central site would have aided performance tuning, debugging or balancing. I have also been involved in research projects where logging and tracking individual test installations was required. This lightweight API provides a very useful set of logging functions without placing major demands on the software developer.

Appendix: WebTics API

In this appendix the technology and Application Programming Interface (API) is documented in more detail. The documentation is intended to be useful for developers who have decided to consider using the system, or wish to see how we have implemented our system to improve their own metrics systems.

Selecting Technology

The API discussed in this chapter is a minimal event and message-passing interface using HyperText Transfer Protocol (HTTP) for all messages. A small library implemented in C++ provides the client-side functionality and a back-end for storing the data is implemented using PHP/MySQL and an Apache server (WAMP/LAMP). Using HTTP for system communication allows easier cross platform integration

and development. The only assumption is that there exists a network connection and network handling within the operating system.

Advantages:

- A simple API provides most of the functionality needed in most projects.
- It does not require the user to write any network routines.
- Text-based message-passing makes debugging easier.
- The front-end client can be developed in any language with networking.

Disadvantages:

- This system does not maintain a permanent direct connection, therefore the messages and replies may not arrive or may arrive out of order. They are time-stamped and can be reordered after-the-fact.
- Requires some knowledge of SQL and PHP for editing the visualization and data mining system.
- Not GUI-based, so harder for designers to use.

How to Start

1. Create a local Database and web server for testing. Download the latest version of a LAMP/WAMP server. (The Uniform Server is a good one currently)
2. Download the metrics library from GitHub. This will give you the latest stable version of the API.
3. Run the SQL build script included in the metrics API download either locally or on a remote server. (If you want to make changes to the default logging system you can edit the setup file).
4. Place the php and html files either in your local root or on the server you are using.
5. Include the header files and the library files in your project. Make sure you follow the instructions for setting up the metrics class, in the README at the top level of the project.
6. Start logging events from your game using the LogEvent calls.
7. View your data using phpmyAdmin or the realtime visualization tools in the visualization path of the metrics site included in the API.

Overview of the API

Using this system requires the inclusion of one C++ header, GameMetrics.h. The system uses a second header file, GameMetricsDefaults.h, containing the enumerated types and labels describing the events to be logged. This header is not included by the programmer, but is loaded as part of the GameMetrics.h file. It may need to be included elsewhere to provide the event types to other

program files. The functionality is provided by adding a single source file, `GameMetrics.cpp`. Alternatively the static library `GameMetrics.lib` may be linked into the project.

What follows is a summary of the usage of the API we discuss. This process is relatively simple as you only need to include a couple of setup methods for identification and tracking of the player and game session. The Game Metrics class has been implemented using the Singleton design pattern. This enables the metrics systems to be accessed from any location in the program, and controls the problem of multiple instantiations.

Creation of the metric system does not establish a connection to the logging system. A session must be opened explicitly. Any `LogEvent()` calls used to log an event before the session is opened will have no effect.

Session Order

We will explain each part of the API based on how you would use the system. The pseudo-code for the general structure of a program setting up a logging session is:

1. Get the instance of the GameMetrics singleton
2. Initialize the metric system
3. [optional] Check Authorization if required with unique ID
4. [optional] Request and set authorization if not already given for unique ID
5. Open metric session with unique ID
6. [optional] Register version number and event types if required
7. [optional] Get game parameters (type 1)
8. [optional] Start play session
9. [optional] Get game parameters (type 2)
10. While (the game is playing)
 - ...
 - Log events
 - ...
11. [optional] Stop play session
12. Close metric session

Get the Instance of the Singleton

This method must be called to create the GameMetrics system. The system has been implemented as a Singleton so there is no public constructor. After creation further calls to `GetInstance()` will return a pointer to the GameMetrics system that has already been created. The calls to `GetInstance()` may be made from any location in the program. If the system has not previously been initialized, then passing `autoInitialise` as `true` will instruct the system to use the default server URL and PHP path defined in `GameMetricsDefaults.h`. If `false` is used the developer must use the `Initialise()` method. The default value is `true`.

```
GameMetrics* metricsSystem = GameMetrics.GetInstance  
(autoInitialise);
```

The addition API calls will be prefixed with `metricsSystem` to reflect their usage.

Initialize the Metric System

If the system has not been auto-initialized, or for any reason needs to be reinitialized this method may be used. The sever URL and the PHP path are passed to the system.

```
metricsSystem->Initialise(serverURL, serverPHPPath);  
authorised = metricsSystem->IsAuthorised(uniqueID);
```

Request and Set Logging Authorization

After the programmer has obtained permission from the user via any mechanism available to them this method is used to update the database. The unique ID for this application is stored with the version number of the application that has been given permission. The version ID string may be empty if authorization per update is not required.

```
metricsSystem->SetAuthorised(uniqueID, trueFalse,  
versioned);
```

Open a Metric Session

This method is called near the start of the program. This sets the timer in the Metric system to zero. All messages sent to the logging system for this session will be time stamped with an offset from this point. At this stage you have to decide whether to track the individual user from session to session. For maximum anonymity, either required by law in a given location or because the user-base will be reluctant to participate, the logging system can be run with a session key linked only to the specific game session. This key allows events within the session to be linked but has no information linked to the user. Over a series of collected data events you will still be able to analyze how the player population behaves but will not be able to trace changes in a specific series of play sessions over time.

The alternative is to send some identifying information to the metric server to be linked to the session ID. There are many options for this. A registered game could sent its serial number for example. A player may be logged into a game environment such as Steam or your own game servers. Alternatively the user may be playing the game within Facebook or a similar social networking system. The availability of identifying information to your metrics system, and your obligation regarding storage and recording will be bound by the terms of service of the overall environment.

The benefit of being able to identify a specific game installation is the tracking of a player over time. The play actions of a group of players with a high win ratio, or

high scores allows you to identify strategies that permit the player to beat the game, and guide where the rule may need to be tweaked for balance or to provide a continual challenge. The metric session ID and play session ID will not be consistently associated with a single player so an installation ID is required.

Further discussion on using the API will assume some persistent identification of a game installation between game sessions.

```
metricsSystem->OpenMetricSession(uniqueID);
```

Register Version Number and Event Types if Required

This method sends all the current event labels to the database with the current version number. There are two forms, one without parameters, which sends the values defined in `GameMetricsDefaults.h` and the other which passes two other user defined arrays of strings to the database.

```
metricsSystem->RegisterEvents();
metricsSystem->RegisterEvents(versionID,
                               eventsArray, numberOfEvents,
                               subeventsArray, numberOfSubevents);
```

Getting the Game Parameters (1)

This is an optional action. If you chose to write your game to have parameters that can be reinitialized during game play you may return a new set of parameters to the game. These parameters may control any aspect of the game, from the points scored for each action to decision thresholds for the AI system or new build cost/times to alter the game balance. A game can receive one of a predefined set of parameters, which is recorded in the metric system. The outcome is observed or the user is questioned about how they enjoyed the play experience. This is of use especially to game designers or developers of serious games.

The text parameter is there only as an option, it may be left NULL. It allows a the current game installation or runtime information to be sent to a more complex back-end system. This information can be used by the back-end system to evaluate which set of parameters should be returned to the game. It is the responsibility of the programmer to manage the memory of this string and delete it after the string has been used.

```
string* parameters = metricsSystem->RequestParameters
(dataString);
string* parameters = metricsSystem->RequestParameters();
```

Start a Play Session

If a game has sets, levels or other well defined breaks in timing or difficulty it is useful to have explicit markers between these stages. Starting a play session places a marker in the metric system that may be used as a delimiter when the metrics are later analyzed.

If the game is a continuous experience then this marker will add no additional clarity to the metrics collected. You may choose to issue `StartPlaySession()` immediately after `OpenMetricSession()` and `StopPlaySession()` immediately before a `CloseMetricSession()` for completeness, or ignore the Start/Stop Play Session events.

```
metricsSystem->StartPlaySession();
```

Getting the Game Parameters (2)

This is a repeat of the game parameter request that can be performed at program start up. It is again optional. Parameters could be updated on a per set/level or chapter basis by requesting updates at the beginning of these sessions, while requests sent as start up would be used for the entire game session. It is possible to use the call multiple times, with major game parameters established for the entire game session at start up, and minor tweaks on either a random basis or determined by player behavior at the start of each chapter of the game.

```
string* parameters = metricsSystem->RequestParameters
(dataString);
string* parameters = metricsSystem->RequestParameters();
```

Logging Game Events

The metrics system uses a single call to log events, where the developer includes the game event data, and the system adds the logging and game IDs, the logging time and sending time timestamps.

The full event method takes seven parameters to offer flexibility in how it is used.

```
metricsSystem->LogEvent(type, subtype,
    xInt, yInt, zInt,
    valueDouble, dataString);
```

There is also a debugging variant of this call. The goal of the debugging version is to support logging during development. The debug version will only send messages when the game is compiled with the `_DEBUG` flag set in the compiler. The variant allows the developer to put event logging in the program for functional testing which can be compiled out on release.⁵

```
metricsSystem->LogEventDebug( type, subtype,
    xInt, yInt, zInt,
    valueDouble, dataString);
```

The developer may implement wrapper methods for any required subset of the parameters which uses the method above with placeholder values to indicate a field is

⁵This is implemented as an immediate return from the `LogEvent()` method in the library without performing a send. There is a small computational cost to this method. If the developer wishes to have a non-debug version completely removed `LogEvent()` method they will need to use compile time code exclusion with `#ifdef ... #endif` compiler directives.

unused. A selection of wrapper methods are already provided by the API. The wrapper methods are also available for `LogEventDebug()` versions. The default value for unused integers and doubles is `-999999`, the default value for an unused string is `""`.

```
metricsSystem->LogEvent (type, subtype,
                        xInt, yInt, zInt,
                        valueDouble, dataString);
metricsSystem->LogEvent (type, subtype, xInt, yInt, zInt);
metricsSystem->LogEvent (type, subtype, valueDouble);
metricsSystem->LogEvent (type, xInt, yInt, zInt);
metricsSystem->LogEvent (type, valueDouble);
```

Type and subtype are developer defined. They are logged in the database and it is left to the developer or designer to define the meaning of the stored values. For clarity in the source code it can be advantageous to use enumerated types for the event type and subtype, e.g.:

The player was killed by a grenade at location 12,20,30 and they sustained 70 points of damage;

```
metricsSystem->LogEvent ( EventTypes::PlayerDeath,
                        EventSubtypes::Grenade,
                        12, 20, 30, 70.0);
```

The player picked up 10 grenades at location 12,20,30;

```
metricsSystem->LogEvent (EventTypes::PlayerWeaponPickup,
                        EventSubtypes::Grenade,
                        12, 20, 30, 10.0);
```

The player respawned at a location;

```
metricsSystem->LogEvent (EventTypes::PlayerRespawn,
12, 20, 30);
```

The player fell 42 m;

```
metricsSystem->LogEvent (EventTypes::PlayerFall, 42.0);
```

Stop Play Session

This should be used to place a marker in the metric system to terminate a defined period of gameplay. After issuing this command the play session is no longer a valid. Closing a play session twice has not additional effect.

```
metricsSystem->StopPlaySession ();
```

Close Metric Session

This method is used to place a marker in the metric system to indicate the game session has terminated normally. The absence of a close event before the presence

of a new open event from the same user may indicate an unscheduled program exit such as a forced quit or program crash. Closing a session twice has no additional effect.

```
metricsSystem->CloseMetricSession();
```

Data Storage

The data collected by the logging system is primarily in a single table in your testing database. The table is set up to have indexing on event-type and sub-type which will aid the most common queries. Given that this API is for small developers and academic research, the speed of modern hardware is quite capable of accessing the volume of data normally collected. If there are problems with the speed of the database, then there would need to be time spent improving the back-end implementation of this system.

The data has been logged with several hierarchical data types which allows rapid and accurate partitioning of the views of the data-set. Firstly the logging sessions have been initialized with a unique ID, and each play session's start and end has also been logged. Within the play sessions, event-type and event-sub-type, which are developer defined and arbitrarily extensible, provide as many game event categories as required.

The system expects to have a network connection but a failure to find a network can be accommodated. If a network is not available, or the `OpenSession()` attempt fails the system can write log events to a file on disc. On a successful connection to the metric server, if a log file exists on disc it is opened and sent to the server with a marker to indicate it was an offline session. The timing of events relative to the start of the logging session is maintained as is the order so the session can still be used for data mining. Finally all events are logged with a time stamp to allow small parts of a game session, e.g. the 30 s prior to a repeatable crash event, or the first 60 s of an emerging dominant strategy to be viewed.

For viewing the data we recommend that repeated queries to the database should be defined as a database view for efficiency. The data is also well suited for access from a web-based Javascript form. This query may also be built up directly in GUI tools such as phpSqlAdmin.

References

- Altman, D. G., & Bland, J. M. (1995). Statistics notes: Absence of evidence is not evidence of absence. *British Medical Journal*, *311*(7003), 311–485.
- Askedal, Ø. (2011). *Are elderly sufferers of dementia able to play a reminiscence game on a tablet device independently?* Gjøvik: Gjøvik University College.
- Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinel, T., et al. (2008). KNIME: The Konstanz information miner. In *Data analysis, machine learning and applications* (pp. 319–326).
- Bostock, M., Ogievetsky, V., & Heer, J. (2011). D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, *17*(12), 2301–2309.

- Charmaz, K. (2003). Grounded theory. In N. K. Denzin & Y. S. Lincoln (Eds.), *Strategies of qualitative inquiry* (2nd ed., pp. 249–291). Thousand Oaks: Sage Publications.
- Derosa, P. (2007). Tracking player feedback to improve game design. *Gamasutra* http://www.gamasutra.com/view/feature/129969/tracking_player_feedback_to_.php. 7 Aug.
- Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). *Player modeling using self-organization in tomb raider: underworld*. Paper presented at the proceedings of the 5th international conference on Computational Intelligence and Games, Milano, Italy.
- Dustin, E. (2002). *Effective software testing: 50 specific ways to improve your testing*. Boston: Addison-Wesley Professional.
- Fields, T., & Cotton, B. (2011). *Social game design: Monetization methods and mechanics*. San Francisco: Morgan Kaufmann.
- Gee, J. P. (2003). *What video games have to teach us about learning and literacy*. New York: Palgrave MacMillan.
- Göbel, S., Hardy, S., Wendel, V., Mehm, F., & Steinmetz, R. (2010). *Serious games for health: Personalized exergames*. Paper presented at the proceedings of the international conference on Multimedia, Firenze, Italy.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1), 10–18.
- Henderson, C. (2008). Managing software defects: defect analysis and traceability. *SIGSOFT Software Engineering Notes*, 33(4), 1–3.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco: Morgan Kaufman.
- Johnston, C. (2012). Diablo 3: Give a knight a shield, and he kicks you from the game. *Ars Technica*. <http://arstechnica.com/gaming/2012/05/diablo-3-give-a-knight-a-shield-and-he-kicks-you-from-the-game/>
- Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B., Pagulayan, R. J., & Wixon, D. (2008). *Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems*. Paper presented at the proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, Florence, Italy.
- Lewis, J. M., Trinh, P., & Kirsh, D. (2011). A corpus analysis of strategy video game play in starcraft: Brood war. In L. Carlson, C. Hölscher, & T. Shipley (Eds.), *Proceedings of the 33rd annual conference of the cognitive science society* (pp. 687–692).
- Mandryk, R. (2008). Physiological measures for game evaluation. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advice from the experts for advancing the player experience* (pp. 207–235). San Francisco: Morgan Kaufmann.
- Pagulayan, R. J., Keeker, K., Wixon, D., Romero, R. L., & Fuller, T. (2003). User-centered design in games. In J. A. Jacko & A. Sears (Eds.), *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications* (pp. 883–906). Mahwah: Lawrence Erlbaum Associates.
- Rabin, S. (2000). The magic of data-driven design. In M. DeLoura (Ed.), *Game programming gems*. Boston: Charles River Media.
- Sawyer, B., & Smith, P. (2008). *Serious games taxonomy*. Paper presented at the Game Developers Conference, San Francisco.
- Sieber, J., McCallum, S., & Wyvill, G. (2009). BallBouncer: Interactive games for theater audiences. In *Chi '09 workshop – Crowd-Computer interaction* (pp. 29–31). Boston.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *WIRED MAGAZINE* (15.09).
- Yannakakis, G. (2012). Game AI revisited. In *Proceedings of ACM computing frontiers conference*.
- Zoeller, G. (2011). MMO rapid content iteration. In *Game developer conference*, San Francisco.

More Resources

- The programming principles used to develop this API are a mix of our own experience from both industry and academia. In addition to the references above we recommend additional reading.
- The code in this chapter uses the singleton design pattern. The seminal book for design patterns is *Design Patterns: Elements of Reusable Object-Oriented Software*, Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, ISBN 0-201-63361-2.
- For an introduction to PHP/MySQL and Database we recommend *Learning PHP, MySQL, and JavaScript*, Robin Nixon, ISBN: 0-596-15713-4.
- Those with some experience may find it useful to read *PHP5 and MySQL Bible*, Steve Suehring, Tim Converse, and Joyce Park, ISBN: 0470384506.
- Purely as a language reference <http://www.w3schools.com/> provides information for all the web related programming languages used in this implementation.
- For research on game usability and testing the work of the Microsoft Studios Research is excellent for both collecting and analyzing game experience.
- The work of Anders Drachen, Alessandro Canossa, and Georgios Yannakakis at the ITU in Denmark is an excellent starting point for work in the research community on game metrics. Particularly the recent work on Game Metrics Mining with IO Interactive. http://game.itu.dk/index.php/Game_Metrics_Mining

Chapter 11

Interview with Darius Kazemi

Magy Seif El-Nasr

Darius Kazemi has over 10 years of experience analyzing game telemetry and metrics from casual games to AAA titles. He is currently the Director of Community Development at Bocoup. Before joining Boucoup, Darius worked as a *Lead Analyst* at Blue Fang Games, later founding and becoming the President of Orbus Gameworks, one of the earliest game analytics companies. He later joined Turbine, Inc. as a *Data Analyst*, and switched to become a *Quality Assurance Contractor* also at Turbine, Inc.

His analytics work span multiple companies, including Popcap working with Facebook games, such as *Bejeweled Blitz* (Popcap, 2010) and NetDevil/Gazillion Entertainment working with Massively Multiplayer Online Games, such as *Lego Universe* (NetDevil, 2010), Turbine, Inc. working with Massively Multiplayer Online Games, such as *Dungeons and Dragons Online* (Turbine, 2006) and *Lord of the Rings Online* (Turbine, 2007), Blue Fang Games working with the Facebook game *Zoo Kingdom* (Blue Fang, 2010), and GameSpy.

At Bocoup – an open web technology company – he creates new open web technologies and helps such technologies become viable through consulting, training and evangelism. He also researches next generation browser technologies, implements software for clients, runs events workshops and training, and contributes to numerous open source projects.

M. Seif El-Nasr, Ph.D. (✉)
PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

Q: what is your background and how did you get into telemetry and metrics analysis?

Darius: My degree is in electrical engineering. I became a data analyst because I was in the QA department and I knew how to program and write sql queries. I heard we had a database in the company, and thus started to do queries and write reports. This, from there on, became my job.

Q: Define telemetry and Metrics and how are they useful for the industry?

Darius: Telemetry and Metrics are the same. They both represent automated data collected from users. Within the industry, the topic of telemetry and metrics, and the terms themselves are very broad. There are a variety of data that people collect, and they are often collected in different formats. For example, some companies collect data in the form of log files or long text files that one will have to parse. Some other companies format the data into databases that one can query. Databases can also take on different forms: relational or non-relational.

In addition, the data is usually encoded differently depending on use. It can be encoded in human readable format. However, sometimes it is not. For example, at GameSpy most of the data is stored in non-human readable format (binary format). This is due to the fact that this type of data is mostly used by the game to display leader board statistics rather than by human analysts.

Q: Are all GameSpy data encoded in a non-human readable format?

Darius: No. However, I would say most of it is recorded in non-human readable format. This is due to the fact that the data is mostly used by and for the game. GameSpy is well known for developing algorithms that use this data for purposes of matchmaking, leaderboards, etc. GameSpy has been around since the mid-1990s and certainly, for the first 10 years the data was collected strictly for non-human use. However, GameSpy recently hired me as a consultant to investigate the utility of the human readable data they have, and thus they definitely have some that are human readable.

Q: What is the utility of the non-human readable data for designers?

Darius: These types of data can be useful for designers in several ways:

- (a) Progression information: by taking a look at leader boards, designers can, for example, find out how long the people have progressed through their games.
- (b) Many companies use achievements as a form of metrics. They will use achievement data to understand: how many players are playing the game, what percentage of them finished level 1 achievements, level 2 achievements, and so forth. This type data is not usually stored to be queried by a data analyst, but rather is structured for the Xbox Live or PSN network, and is really designed for algorithmic use. However, it can still be useful for designers.

Q: Can you give me an example of how the achievement data have been used?

Darius: yes, mostly it is used to determine issues of progression. Every game is divided into several achievements. This is how most game companies know what percentage of the game was finished by what percentage of the users. For example, they can look at this data and say: 10% of our players received their end of game achievement.

Q: Do designers usually tune the game based on that?

Darius: well, that depends on the company and the kind of game we are talking about. For most console games, it is probably too late to tune the game when they get this type of data. By the time those games are out in the wild, it is too late to do much to tweak them. However, for persistent games, like an MMO, where they plan to release patches, designers often plan to use the data and analysis to release modifications to the game based on what analysts find out from such data. However, for most console games, this data can still be useful to inform their next titles.

Q: You have worked with persistent type games; can you give me examples of the use of metrics within these kinds of persistent games?

Darius: There are many types of data that online games collect. One category of data that companies collect is player demographic data; specifically, data about player avatars and what they are doing: what items do players have in their inventory, what level is the player at, what types of content do they like to interact with, etc. To give you a concrete example, in *Lord of the Rings*, we would collect information to know that a certain percentage of the characters on a particular server play a particular race y and are at level z of that character race. This was an important measure of progression, because in these games, you have a leveling system, and thus one can tell how many players are at any particular level, which then can convey a sense of progression of the player population.

In addition to avatar data, event-based data is also collected and is useful for tracking when events, like combat, are resolved and how they are resolved. This gives you much information about your players and the game. You then can take such aggregate data and break it out by variable or dimensions:

- For example, if your game has combat, most analysts will log every time a combat is resolved, logging specifically how the combat was resolved, e.g., whether it was a win or loss or a death for the player, if the player didn't die, how long it took him to win, what sort of monsters he faced, etc.
- You can also tie the event to where it happened in the world (if the game has spatial metaphor), and thus build heatmaps.
- You can also look at the numbers, break it out by player level and figure out: what is the win/loss ratio of combat for players in each level.
- You can also break this data out by class, or other variables.
- And so on.

Q: How do designers use this type of analysis?

Darius: I would say that since 2009, it has become standard procedure for persistent online games to adjust the game based on this type of analysis. We can take a clear example to show how designers can use such analysis to adjust their game. Let us assume that one can analyze metrics data to pick out anomalies. For example, if every player spends 20% of their time in combat, but a particular class of players spends 50% of their time in combat. This information is then given to designers. The first thing a designer would do with such anomalies is ask if these behaviors are intended. If, for example, this *is* the combat class, then that would be classified as a

perfectly normal behavior. If it is not, then designers will need to investigate further by drilling down in the data some more. Perhaps there is a subclass that is spending more of their time in combat, and thus skewing the data. Another option for designers to do is to step back from the data and look at the design more holistically. I have seen designers also drop into the game as administrators and try to view the play sessions of these people, who are exhibiting the anomalous behavior, to see what is happening. Through all these investigations, designers then determine if this behavior is desirable or a change in the design is necessary. Designers often want to create normal curves, but such curves may not be beneficial for the players or the game. There may be a reason why some players are exhibiting anomalous behaviors and designers may want to encourage that.

Moreover, in some cases, you may want to investigate how the group of people who exhibits the anomalous behavior monetize vs. others. It may be the case that players who are spending all their time in combat (considered anomalous) are the ones spending more money. Therefore, perhaps it is beneficial for the game developers to put more resources into the amount of time people spend in combat (i.e. encouraging the anomalous behavior), because there might be something compelling about it.

Q: The design team is then responsible for figuring out the strategy of what to do?

Darius: It is the design team, but when monetization comes into the picture, the business team will be more involved.

Q: Are there any standardized techniques for figuring out these types of anomalies?

Darius: No. There aren't. There are no standard techniques in the industry, yet. The industry is still young. There are very few practicing analysts, and we haven't shared knowledge or worked together as a community yet.

I have used specific strategies in the past, and I can provide you with some examples that I found useful. One strategy is to work with designers to figure out the thresholds for the key variables that we are collecting for the game. For example, designers may indicate that a particular dungeon should take more than 10 min to finish, and thus if a player spends less than 10 min to finish the dungeon then it is a red flag, i.e. there is something wrong. Another strategy that worked for me in the past is to use data clustering and statistical analysis to reveal outliers. For example, you may check the data and see that 10 min to finish that particular dungeon is actually the norm. Therefore, you will have to change your values; perhaps you will care more about anyone who finishes the level in less than 2 min. Designers often have an intuition about how things should be, and they are usually right, but not always. Sometimes the data proves that they are wrong and you will have to reset their expectations.

Q: How is the metrics process different for the different types of games that you worked on? Does the business team have more to say in the analysis of online games than in console games, for example?

Darius: Yes. If the console game is a persistent console game, then it is probably closer to an online social game, like an MMO or a Facebook title. If it is a single player console game, or an online game that is not persistent, then there usually will be fewer metrics collected, because for such a title, the company will probably patch the game once or may be twice depending on the life cycle of the game. Generally speaking, for the non-persistent games, you are mostly looking at achievements. Some companies or most companies will use a service, like GameSpy. There is a big difference between the number of metrics collected for the different types of games; I would say in the order of 10s of individual metrics for non-persistent as opposed to 100s of metrics for a persistent game.

Q: What are the major surprises that you found when working with metrics?

Darius: Surprises are rare. I was surprised early on in my analysis career with how problems can be amplified on forums. Several times we run into problems where if you read the forums you would think that every single person in the game was having this problem, but then when you look at the numbers, you find that there is only 5–10 people who are facing that problem.

There were some pricing decisions that we made that resulted in unexpected behaviors, but if you know your economics you would know that such behavior was perfectly expected. In *Zoo Kingdom*, the primary way we made money was selling Zoo animals. There were two types of animals: free and premium. The premium animals were priced at 30 cents each. At some point in time, we introduced few animals at the cost of \$1.50, expecting that they wouldn't sell. However, we made a lot of money from these animals. People were buying the \$1.50 animals at the same frequency they were buying the 30 cents ones. I have seen so many articles online that suggested that that would be the case, but it was still hard to convince the company to go that route without actual numbers.

Q: What do you think is currently missing in terms of the field, especially with online games, and where do you think we should be heading?

Darius: [pause]. I will talk about why I am no longer in this field, and perhaps this can suggest some lessons towards future growth. As an analyst working with designers is great; they understand the limitations of the data and the methods, and thus are less likely to jump to conclusions. On the contrary, working with the business side is problematic.

Business teams are interested in questions, such as ‘what is the lifetime value of the user [LTV, see Chap. 4]?’ This is an industry standard metric. It describes the amount of money that you would expect to make from a given user in their lifetime playing the game. Early on in a game's life cycle you cannot know the answer to this question. This is due to the fact that you simply do not have enough data to make an accurate estimate. However, after the game has been out for a year, you will have enough data to generate an estimate. The way you would do this is to first figure out how much time players play. We can assume, for example, that in your sample: 99% of the players play less than a year. Given this window of time, we can then determine the length of time they play. We can assume that, in this case, the average player

plays for 4 months before they stop playing. We then calculate how much money we get on average from players over the course of these months. This figure can then give you the user lifetime value, e.g., if players end up paying \$12.00 within the 4 months, then their lifetime value is \$3/month and lifetime lasts 4 months.

There is a major issue here. Over the course of this time window the game has been changing, and thus you cannot sample from the whole window. You can possibly sample from a single 2-week period when the game is stable. The other question is: who do you sample? If you do sample within the 2-week period, how can one argue that this sample is representative? It is almost like an incompleteness problem. You cannot define these variables, because they are always changing. Thus, there are many limitations to the estimate that you can provide as an analyst. However, the business teams usually do not understand all these limitations; they need the figures in order to do their job.

Nevertheless, there are some statistics that are truly meaningful, such as the amount of money you make per month or average revenue [ARPU, see Chap. 4]. These statistics are meaningful, because you have the facts and you can provide a true and useful figure. Also, comparing average revenue per user between games is useful. However, when you start estimating the future, the figures that analysts can provide become inaccurate.

Q: So, how can we resolve this problem?

Darius: We have to educate the business people. That is part of the reason I am not doing this anymore.

However, there is another reason I stopped working on this. Working on analysis of metrics for games became too much of the same thing. One usually starts with defining what data to record; this is usually 90% similar between projects, especially between games of the same type or genre. For example, data collected by *Lord of the Rings Online* is very similar to the data collected by *Kung Fu Panda World* (DreamWorks Animation, 2010); they are very different games for very different audiences, but from the data side, analyst collect more-or-less the same data. In addition, the analysis side is also 75% similar from project to project, mostly involved with progression, detecting anomalies, collecting information on spending activities, etc. Only about 30% of the work is intellectually challenging. This is when designers identify a new feature, and we are asked to analyze how users are accepting it. This is the fun and challenging part of being an analyst. However, after 6–7 years of working in this field, I was ready for something new.

Q: Is there a time where the data completely failed to capture what you were trying to get at?

Darius: Here is a story where it was simply impossible to use data to solve the problem we wanted to solve. We talked about combat encounters earlier. Once the combat is resolved you usually record information about the combat, e.g., place, outcome, etc. Imagine you have a freeform game where it is impossible to know when the combat began or when it ended. For example, in a game like *World of Warcraft* (Blizzard, 2004), it is simple to record these events because you know exactly when combat happens. The player walks up and presses attack.

This then constitutes a combat start event. Once all the enemies are defeated or when the player is dead, we can record that the combat has ended. Therefore, there is a clear indication of an event when the combat is initiated and when the combat has ended.

I worked on a game called *Jumpgate Evolution* (NetDevil, not released yet). It was a freeform game with arcade style controls, where you fly around space in a ship, accomplishing missions. Due to its freeform nature, we could not figure out a satisfactory way of indicating when a combat was initiated or when it ended. This is due to the fact that you can fire your gun anywhere at any time, and thus, in some sense, you are always in combat. Such continuous spaces are hard to collect event-based data from. By space, I do not just mean physical space simulation, but design space where there is no clear point to setup an event trigger for metrics collection.

Q: What advise you would give developers who are beginning to use telemetry?

Darius: It is a growing field that is direly needed. I usually get an email once a week asking me to recommend an analyst for one job or another. I believe the field is growing and there are many opportunities.

In terms of knowledge and skills, I would recommend that analysts have both a design and business background in addition to the core analysis knowledge, because analysts often have to work with both groups. Also, analysts need a good understanding of the technical side, at least an understanding of how SQL and databases work and how to build a query. That is one of the things that Zynga looks for, for example. If you are involved in the production phase, you will have to architect the data.

Q: What is next for you?

Darius: These days I am working on HTML 5 game technology. I help people port their games to HTML 5. Additionally, I build games and evaluate game engines. Generally, I am evangelizing HTML 5 to the game industry. I also run conferences around the topic.

About the Editor

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. Magy worked collaboratively with *Electronic Arts*, *Bardel Entertainment*, and *Pixel Ante*.

Part III

Game Data Analysis

This part of the book intersperses theory-heavy contributions with in-depth case studies, examining strategies for selecting the most relevant game variables and then discussing methods for data analysis from datamining to spatial analyses. The case studies come from advanced analysis techniques employed at Crystal Dynamics, IO Interactive and Milestone.

This part has the following take-aways:

- Provide an overview of the forefront of large-scale multidimensional datamining techniques geared to improve user experience.
- Define a common terminology and discuss the process of isolating relevant game variables and turning them into measures for high-level, player behavior modeling.
- Show how industry and academia can successfully partner for complex analyses.
- Demonstrate how game telemetry systems can be successfully deployed in existing game user research practices.
- Describe current practices in spatial game analytics.

The part will consist of five chapters:

- Chapter 12: *Game Data Mining* introduces the area of data mining and provides example data mining algorithms that can be used for analysis of game telemetry. The chapter also shows the concepts through case studies done in collaboration with the industry. The chapter is a contribution from Andres Drachen (co-editor of the book) and Christian Thureau, CTO of Game Analytics a company formed to deliver game analytics services to game developers, Julian Toeluius, Associate professor at ITU, Copenhagen, Georgious Yannakakis, Associate professor at University of Malta, Malta, and Christian Bauchhage, professor at University of Bonn, Germany.
- Chapter 13: *Meaning in Gameplay: Filtering Variables, Defining Metrics, Extracting Features and Creating Models for Gameplay Analysis* investigates the concepts behind telemetry, metrics, and variables collected to measure player behavior and modeling.

- Chapter 14: *Gameplay Metrics in Game User Research: Examples from the Trenches* discusses a case study of the use of game metrics analysis in collaboration with IO Interactive. The chapter is authored by Anders Drachen and Alessandro Canossa (Co-editors of this book) with Janus Rau Møller Sørensen, a user research manager at Crystal Dynamics and IO Interactive, worked on titles including *Hitman Absolution* and *Deus Ex: Human Revolution*.
- Chapter 15: *Interview with Aki Järvinen from Digital Chocolate* is an interview with Aki Järvinen, creative director and competence manager at Digital Chocolate. The interview delves into the use of analytics at Digital Chocolate and its role and importance within the company.
- Chapter 16: *Better Game Experience Through Game Metrics: A Rally Videogame Case Study* shows another case study of using game analytics for a racing game developed at Milestone, Italy. The chapter is a contribution from Pietro Guardini, games user researcher at Milestone contributed to several titles, including *MotoGP 08*, the *Superbike World Championship (SBK)*, and Paolo Maninetti, senior game programmer at Milestone, worked on titles such as *MotoGP08* and the *Superbike World Championship (SBK)*.

Chapter 12

Game Data Mining

**Anders Drachen, Christian Thureau, Julian Togelius,
Georgios N. Yannakakis, and Christian Bauckhage**

Take Away Points:

1. The data revolution in games – and everywhere else – calls for analysis methods that scale to with dataset size. The solution: game data mining.
2. Game data mining deals with the challenges of acquiring actionable insights from game telemetry.
3. Read the chapter for an introduction to game data mining, an overview of methods commonly and not so commonly used, examples, case studies and a substantial amount of practical advice on how to employ game data mining effectively.

A. Drachen, Ph.D. (✉)

PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark

e-mail: andersdrachen@gmail.com

C. Thureau, Ph.D.

Game Analytics, Refshalevej 147, Copenhagen, K 1422, Denmark

J. Togelius, Ph.D.

Center for Computer Games Research, IT University of Copenhagen,

Rued Langgaards Vej 7, Copenhagen, S 2300, Denmark

G.N. Yannakakis, Ph.D.

Department of Digital Games, University of Malta, Msida MSD 2080, Malta

Center for Computer Games Research, IT University of Copenhagen,

Rued Langgaards Vej 7, Copenhagen, S 2300, Denmark

C. Bauckhage, Ph.D.

Fraunhofer Institute Intelligent Analysis and Information Systems IAIS,

Schloss Birlinghoven, 53754 Sankt Augustin

Bonn-Aachen International Center for Information Technology B-IT Dahlmannstraße 2

D-53113 Bonn, Germany

12.1 Introduction

During the years of the Information Age, technological advances in the computers, satellites, data transfer, optics, and digital storage has led to the collection of an immense mass of data on everything from business to astronomy, counting on the power of digital computing to sort through the amalgam of information and generate meaning from the data. Initially, in the 1970s and 1980s of the previous century, data were stored on disparate structures and very rapidly became overwhelming. The initial chaos led to the creation of structured databases and database management systems to assist with the management of large corpuses of data, and notably, the effective and efficient retrieval of information from databases. The rise of the database management system increased the already rapid pace of information gathering.

During later years, a virtually exponential increase in the availability of data is emerging in fields across industry and research, such as bio-informatics, social network analysis, computer vision and not the least digital games. Today, far more data is available than can be handled directly: business transaction data, scientific data from, e.g., telescopes, satellite pictures, text reports, military intelligence and digital games (Berry and Linoff 1999; Han et al. 2005; Larose 2004; Kim et al. 2008; Isbister and Schaffer 2008; Mellon 2009; Drachen and Canossa 2009; Bohannon 2010).

Modern digital games range from simple applications to incredibly sophisticated information systems, but common for all of them is that need to keep track of the actions of players and calculate a response to them, as is discussed in most chapters in this book. In recent years, the tracking and logging of this information termed telemetry data in computer science – as well as data on technical performance of the game engines and applications themselves – has become widespread in the digital entertainment industry, leading to a wealth of incredibly detailed information about the behavior of – in the case of major commercial titles – millions of players and installed clients (Mellon 2009; King and Chen 2009; Drachen and Canossa 2011).

Applied right, **game telemetry** can be a very powerful tool for game development (Kim et al. 2008; Isbister and Schaffer 2008; King and Chen 2009). Not only for analyzing and tuning games, QA, testing and monitoring infrastructure (Mellon 2009), figuring out and correcting problems and generally learning about effective game design, but also to guide marketing, strategic decision making, technical development, customer support, etc. However, it is generally far from obvious how to employ the analysis (Kim et al. 2008; Mellon 2009): what data should we record, how can we analyze it, and how should it be presented to facilitate effect transformation of raw data to knowledge that if fully integrated into the organization?

Narrowing the focus to **behavioral** game telemetry, i.e. telemetry data about how people play games (Chap. 2), there are a wide variety of ways that this kind of game telemetry data can be employed to assist a variety of stakeholders (as

discussed in Chap. 3), during and following the development process, e.g., informing game designers about the effectiveness of their design via user modeling, behavior analysis, matchmaking and playtesting, something that is evident from the range of publications and presentations across academia and industry, including: Kennerly (2003), Hoobler et al. (2004), Houlette (2004), Charles and Black (2004), Thureau et al. (2004, 2007, 2011), Ducheneaut and Moore (2004), DeRosa (2007), Thompson (2007), Kim et al. (2008), Isbister and Schaffer (2008), Thawonmas and Iizuka (2008), Thawonmas et al. (2008), Coulton et al. (2008), Drachen et al. (2009, 2012), Missura and Gärtner (2009), Williams et al. (2009, 2008), Thureau and Bauckhage (2010), Pedersen et al. (2010), Yannakakis and Hallam (2009), Weber and Mateas (2009), Mellon (2009), Seif El-Nasr and Zammitto (2010), Seif El-Nasr et al. (2010), Thureau and Drachen (2011), Yannakakis and Togelius (2011), Moura et al. (2011), Erfani and Seif El-Nasr (2011), Drachen and Canossa (2011), Yannakakis (2012), Bauckhage et al. (2012) and Gagné et al. (2012).

There is a wealth of information hidden in good game telemetry data, but not all of it is readily available, and some very hard to discover without the proper expert knowledge (or even with it). This has led to much game telemetry data being tracked, logged and stored, but not analyzed and employed. The challenge faced by the game industry to take advantage of game telemetry data mirrors the general challenge of working with large-scale data. Simply retrieving information from databases – irrespective of the field of application – is not enough to guide decision-making. Instead, new methods have emerged to assist analysts and decision makers to obtain the information they need to make better decisions. These include: automatic summarization of data, the extraction of the essence of the stored information, and the discovery of patterns in raw data. When datasets become very large (we consider any dataset that does not fit into the memory of a high-end PC as large-scale, i.e. several GB and beyond) and complex, many traditional methodologies and algorithms used on smaller datasets break down. In fact, any algorithm that scales quadratically with the number of samples is not feasible to use on large data. Instead, methods designed for large datasets must be employed. These methods are collectively referred to as **data mining**. Data revolutions call for novel analysis methods that scale to massive data sizes and allow for effective, rapid analysis, as well as results that are intuitively accessible to non-experts (Han et al. 2005; Larose 2004).

Using data mining methods in the context of game telemetry data – what we will here call game data mining – we can for example:

- Find weak spots in a games' design (Chap. 7; Thompson 2007; Kim et al. 2008; Drachen and Canossa 2009; Gagné et al. 2012)
- Figure out how to convert nonpaying to paying users (Chap. 4; King and Chen 2009)
- Discover geographical patterns in our player community,
- Figure out how players spend their time when playing (Chaps. 18 and 19, DeRosa 2007; Moura et al. 2011; Drachen et al. 2012)
- Discover gold farmers in an MMORPGs (Thawonmas et al. 2008),

- Explore how people play a game (Chap. 14; Drachen and Canossa 2009),
- How much time they spend playing (Williams et al. 2009)
- Predict when they will stop playing (Mahlman et al. 2010; Bauckhage et al. 2012)
- Predict what they will do while playing (Weber and Mateas 2009)
- Which assets that are not getting used (Chap. 14)
- Develop better AI-controlled opponents or make games that adapt to the player (Charles and Black 2004; Thurau et al. 2007; Missura and Gärtner 2009; Pedersen et al. 2010; Yannakakis and Hallam 2009)
- Explore and use of social grouping (Thurau and Bauckhage 2010) – and much, much more.

This chapter will outline how large-scale data mining can be effectively carried out on game telemetry data (i.e. telemetry from game clients/game servers, which can include data on players), and cover a range of scenarios, from behavior analysis of individual players and how they give rise to patterns, to interpretation of larger scale structures like guilds in massively multiplayer online games. To begin with, we will provide an introduction to data mining in general and game data mining specifically, good data mining practices and methods, as well as notes on tools and challenges to game data mining. This should by no means be viewed as a thorough introduction to data mining – that requires an entire book. Fortunately such books exist, for example Han et al. (2005) is a good starting place for the novice data miner.

Following, the major categories of data mining will be outlined and a number of case studies used to exemplify some of the commonly used game data mining approaches, covering supervised and unsupervised methods, with examples from, e.g., *World of Warcraft* (Blizzard, 2003), *Tomb Raider: Underworld* (2008, Eidos Interactive) and *Heroes of Newerth* (2010, S2 Games), as well as case examples obtained from concrete production contexts. As the interpretability of data analysis results is important, we focus on methods that go well beyond descriptive techniques to provide more meaningful, useful and actionable data representations, enabling the analysis of player behavior across millions of individuals.

On a practical note, the first half of this chapter (Sects. 12.1 and 12.2) is written for the general audience and does not require previous knowledge of data mining. The second half (Sects. 12.3 and 12.4), which focus on case examples of different data mining methods, use however data mining terminology, includes some use of formulas, and some sub-sections may require knowledge of statistics and dimensionality reduction methods. Section 12.5 takes a specific look at online games, including Free-to-Play (F2P), as these have received particular attention with respect to data mining.

12.2 Data Mining in Games Background and Overview

In this section, data mining is introduced and an overview of the main types of methods presented, leading up to the subsequent sections, which will cover the specific methods employed for analyzing game telemetry data.

12.2.1 What Is Data Mining?

Data mining is a somewhat nebulous concept, and there is no single definition of what it is (Chen et al. 1996; Fayyad et al. 1996; Han et al. 2005). The name itself – data “mining” – is derived due to the similarity between searching for valuable information in large databases and mining rocks for valuable ore. Both imply either sifting through large amounts of material randomly or use intelligent methods to pinpoint where the valuable material is. The term is something of a misnomer though, as the goal of mining data is not data, but knowledge (i.e. “knowledge mining”). However, data mining sounded sexier and became the accepted term, overshadowing other terms such as knowledge discovery, knowledge extraction and pattern discovery, which better describe the complete process. According to the Gartner Group (www.thegartner-group.com), data mining is: “*the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques.*” Similar to this definition, others usually emphasize the exploration and analysis of large quantities of data, in order to discover meaningful patterns. Data mining forms an amalgam of methods and ideas, drawing inspiration from different fields of science and business, including machine learning, artificial intelligence, pattern recognition, medical science, statistics and database systems – in many ways, data mining has emerged in the space between these fields of research (Berry and Linoff 1999).

Depending on the definition, data mining is either a step in or the whole of, the process of knowledge discovery in databases (KDD), a concept originating from 1989, referring to the non-trivial extraction of implicit, unknown and potentially useful information from data in databases (Berry and Linoff 1999). It used to be that KDD was viewed as the overall process of discovering useful knowledge from data, while data mining was the application of particular algorithms to extract patterns from data without the additional steps of the KDD process, but the difference is largely academic.

The same is the case for the difference between data mining and statistics. Many data mining methods stem from statistics, and data mining itself largely arose due to the need in statistics to adopt and invent new ways of working with huge masses of data. Over the 1990s, working with large masses of data became synonymous with data mining, although the methods applied could also be called statistics. At the mathematical level, there are various arguments that can be leveraged, but the main difference again relates to whether data mining and statistics refers to specific methods, or the entire process of working and extracting meaning from data. If statistics is viewed as a set of methods, and data mining the entire process, they are different – but again, in practice, from the trenches of game analytics, there is not a lot of difference.

Irrespective, there is a very strong human element in data mining. Early definitions of data mining emphasized automatic or semi-automatic methods, but it is important to note that the human element is vital when exploring and analyzing large datasets – data mining has become synonymous with automatic techniques, which has led people to believe that it is a product that can be bought. There is a variety of black box software available on the market, which embeds powerful algorithms, making their misuse proportionally more dangerous. As with any other technology, data

mining is easy to do badly. Analysts may apply inappropriate algorithms to data sets that call for a completely different approach, for example, or models may be derived that are built upon wholly specious assumptions. Therefore, an understanding of the statistical and mathematical model structures underlying the software is required.

Data mining is a discipline. In our view, humans need to be involved at every phase of the data mining process. In a game development context, at least in some steps, those humans need to be the people who design, code, test and ultimately play, games. There is no quick fix for a game developer wanting to employ data mining on e.g. user telemetry data, however, as noted by Larose (2004), purely explorative methods can be quite powerful in their own right, and require much less training and specialized knowledge than semi-automatic or automatic processes, although a general understanding of the data is always required.

12.2.2 The Knowledge Discovery Process in Data Mining

It can be tempting to approach data mining haphazardly, without an overall strategy. Therefore, a cross-industry standard was developed in 1996, which took an industry-, application- and tool-neutral approach to data mining, defining the *Cross-Industry Standard Process for Data Mining* (CRISP-DM) (www.crisp-dm.org) (Chapman et al. 2000). CRISP represents a fundamental good approach to data mining processes, and various specialized variants exist aimed towards particular industries or problems. It can seem a bit cumbersome to apply the full CRISP process to each and every question that we want answered via game telemetry, and in practice, some of the phases will be very quick to go through, especially if the analysis has already been performed on a previous version of a game or earlier user-behavior dataset. However, CRISP provides a non-proprietary and freely available standard process for fitting data mining into the general problem-solving strategy of a business or research organization. It is an iterative process, fitting well into the typical agile and rapid-iterative approaches applied in game development (Mellon 2009). The phase sequence of CRISP is adaptive – i.e., the next phase in the sequence of six defined phases depends on the outcomes association with the preceding phase. For example, it may be necessary to return to the data preparation phase for refinement before moving on with the modeling phase – it is a typical situation that the solution to a question leads to further questions, not the least when working with player behavior in games. Importantly, lessons learned from past projects should be brought to bear as input into new projects.

In the context of game development, the data in question can originate in game telemetry (data from installed game clients) or any of the traditional sources of business intelligence, e.g. production and marketing (Romero 2008; Mellon 2009; Drachen and Canossa 2011). The CRISP phases are as follows (modified from: www.crisp-dm.org, and Larose (2004)):

1. Business/research understanding

- Define the project objectives and requirements.

- Translate the goals and restrictions into the formulation of a data mining problem definition.
- Prepare a preliminary strategy for achieving these objectives.

2. Data understanding

- Collect the data (extract the data from a database).
- Use exploratory data analysis (EDA) to familiarize yourself with the data and discover initial insights.
- Evaluate the quality of the data.
- If desired, select interesting subsets that may contain actionable patterns.

3. Data preparation (typically the most time-consuming phase)

- Prepare from the initial raw data the final data set that is to be used for all subsequent phases.
- Select the cases and variables you want to analyze and that are appropriate for your analysis (this is sometimes performed after transformation and cleaning).
- Perform transformations (or consolidation) on certain variables, if needed. Under transformation, the selected data are transformed into the form needed to perform the mining procedure in question, e.g. normalizing values.
- Cleaning: Clean the raw data (remove noise and irrelevant data) so that it is ready for analysis.

4. Modeling

- Select and apply appropriate data mining technique (modeling, exploration, synthesis etc.). Different techniques may be used for the same problem.
- If the technique results in a model (most do outside of explorative analysis), calibrate model settings to optimize results.
- The process can be repeated with new selections and transformations of the data, gradually refining the result or integrating new relevant data sources, in order to get different, more appropriate/valuable results.

5. Evaluation

- Evaluate the results and/or models delivered for quality and effectiveness.
- Check that the model in fact achieves the objectives set for it in the first phase.
- Check whether some important facet of the problem has not been accounted for sufficiently.
- Come to a decision regarding use of the data mining results.

6. Deployment

- The discovered knowledge is presented to the relevant stakeholder (designers, producers, marketing, management), using a choice of knowledge representation, e.g. a visualization or report. E.g., a game telemetry analyst develops a heatmap of a multi-player shooter level to present to the designer of that level (see also Chap. 17).
- Make sure the presentation is done in such a way that the target stakeholder can understand. Explain the result in a way that helps the target stakeholder to understand, interpret and act upon the data mining results.
- The target stakeholder carries out deployment within the organization.

12.2.3 Database Navigation

When dealing with data stored in databases – irrespective of the format – there are three fundamental **navigational tasks** that the chosen database format must allow: drilling down (vertical), drilling across (horizontal), and controlling time. The latter is the most obvious: we have to be able to select data from only a specific build, a particular user test, or segment of time. We mention these here because basic navigation of game telemetry data is often the first step needed in data mining, and sometimes the only step needed to answer questions.

Drilling (or navigating) is a method for interactively navigating or exploring data horizontally and vertically in the dimensional structure of a database.

Drill-across (or drill-through) **navigation** occurs across multiple dimensions (used commonly with OLAP, Online Analytical Processing, a class of functions that enable users to dynamically and in real-time analyze and navigate data, e.g. in a data cube), and is used for e.g. comparing different variables for a specific dimension, for example playtime and item purchases for all players from Europe. Similarly, identifying the top ten most profitable players in a free-to-play game from each game server, is an example of a drill-across analysis. Drill-across navigation can operate across dimensions, measures or attributes in OLAP and data warehouses.

Drill-down navigation is a means for exploring data in greater detail (more low-level) than is currently displayed (Kim et al. 2008). The term drill-down is commonly encountered in game data mining contexts. This is because drill-down analysis is a method for in-depth analysis of data, which makes it very useful to especially player behavior analysis, where the root causes of behavioral patterns are often nestled deep within the data – e.g. a specific checkpoint missing, a single mob being too powerful, a pathway between to areas going unnoticed (Kim et al. 2008; Drachen and Canossa 2011).

For example, viewing aggregated completion times across ten levels in a game, and noting that level 5 completion times are very high, drilling down would then be to look at the data for pertaining to level 5 only for each player. This kind of process is commonly used in game analytics to locate the root causes of an observed effect. See e.g. Romero (2008), Kim et al. (2008), Drachen et al. (2009) and Chap. 14 for examples.

To take an example (Fig. 12.1): A game developer considers a simple breakdown of data consisting of a few variables, here the average completion times for the levels of a game (five levels). At this top level, he notices that a level appears to take longer to complete than others (level 4). This is not intended by the design, and could therefore indicate a problem. In order to explore why, the underlying data need to be exposed, in this case, a breakdown of the completion times for individual components of the level. In this more detailed view of the data, it may be noticed that a specific sector of the level is where the players spend a lot of time (Main Hall). Drilling down further into the metrics data captured from the level, it may be found that the root cause is that players have trouble beating a specific type of enemy and keep dying (Evil Conductors – they stamp your ticket really, *really* hard), whose difficulty can be adjusted to accommodate. If the cause of the observed pattern is not obvious from looking at the data, it can be

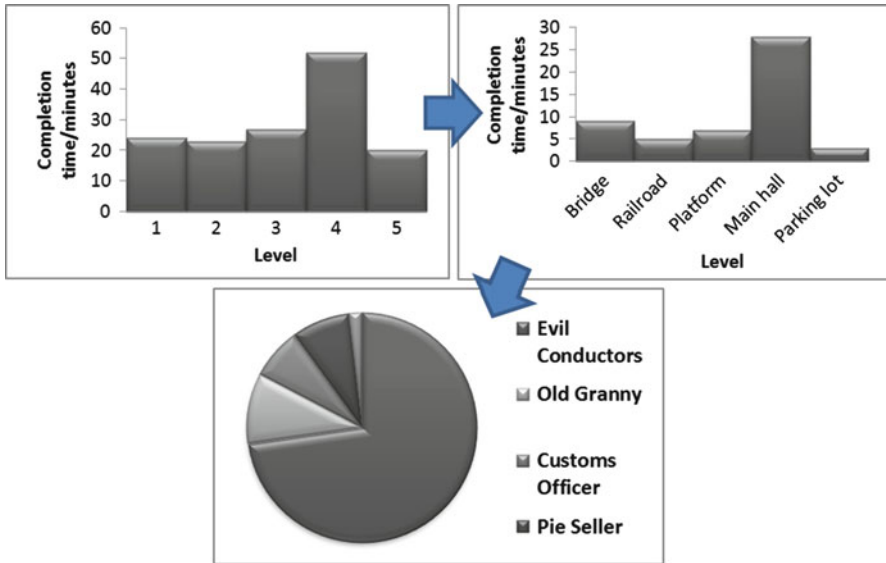


Fig. 12.1 An example of drill-down navigation from a fictive game. See text for explanation

Six Myths About Game Data Mining

1. *With game data mining, we can fire our designers as the testers/users will tell us what they want! We will save heaps of money!* Wrong: mining gameplay telemetry data is incredibly useful for evaluating and testing design, but telemetry data cannot tell you what design is, nor how your players feel or if they have a good experience – game data mining is not a replacement for good game design.
2. *With game data mining, we do not need user testing! We can fire our testing department and save heaps money!* Wrong: Game data mining goes hand in hand with user-oriented testing and -research, but does not replace it. With mining of gameplay metrics data, a powerful supplement to playtesting, usability testing and physiological testing is gained.
3. *Game data mining is autonomous, requiring little human oversight! The tools are automated; we just turn them loose on our data and find the answer to our design/business/marketing problem! We can fire our business analysts and save heaps of money!* Wrong: There are no automatic tools that will solve your problems – data mining is a process, as highlighted by CRISP. Additionally, data mining requires significant human interactivity at each phase, and also for the subsequent quality monitoring.

(continued)

(continued)

4. *Game data mining pays for itself quickly! Let us invest in tools, infrastructure and people right away and save heaps of money!* Wrong: well, partly wrong. The return rates vary, depending on the specific situation, the game, the size of the developer or publisher, etc. The return will be there in terms of improved knowledge, but careful planning needs to go into deciding on an initial setup and the strategic and practical goals.
5. *Game data mining will identify the causes of all our problems! We will make heaps of money integrating game data mining in our business!* Wrong: the knowledge discovery process will help identify and uncover patterns of behavior in the data whether user-derived or business-derived, and these can be highly valuable, but it requires human interpretation to identify the causes of the patterns (with the help of analysis).
6. *We need to obtain data on everything! Data equals value, we will make a heap of money!* Wrong: you need the right data, to solve the problems you have. Just measuring everything will waste resources. Getting the right data requires as much thought as their analysis.

useful to consider the actual game environment – e.g. finding that players do not spot the big weapon they needed in the room preceding the boss. Drill-down analysis work in this way to identify the root cause of patterns that emerge much “higher” in the data. At the lowest level of a drill-down analysis are the raw data.

There are a number of other fundamental actions that users should be able to take in a database system, e.g. filtering, sorting, morphing etc., but a full discussion is out of scope here – the reader is referred to Han et al. (2005) and Larose (2004) for additional information.

12.2.4 Separating Gold from Rock in Data Mining Results

When running a game data mining analysis, for example a classification analysis of the behavior of players during development of a game, there will typically be more than one result. So how do we know which one to pick?

Data mining allows the discovery of patterns in the data, and there may be quite a lot of them if the dataset is big enough and heterogeneous enough. Finding the best, most useful and interesting pattern is not always straight-forward. In order to choose the best pattern (or result), we need to be able to evaluate the patterns based on how “interesting” or useful, they are to the specific situation. There are three approaches that can be employed:

Objective measures: these are based on the war data themselves, e.g., validity of the patterns when tested on new data with some degree of certainty. They are by far the easiest to employ and provide hard numbers to evaluate results.

Subjective measures: these are based on the data and the user of the data. As noted by (Geng and Hamilton 2006) *“to define a subjective measure, access to the users domain or background knowledge about the data is required. This access can be obtained by interacting with the user during the data mining process or by explicitly representing the users’ knowledge or expectations.”* Novelty or understandability of a result is an example or a subjective measure of interestingness.

Semantic measures: considers the semantics and explanations of the patterns. A good example is “utility” or “actionability” as an evaluation mechanism.

Identifying and measuring the interestingness of patterns is essential for the evaluation of the mined knowledge and the data mining process in general. Concretely, interestingness measures are useful because they can be used to: (1) prune uninteresting patterns during the mining process so as to narrow the search space and thus improve mining efficiency; (2) rank patterns according to the order of their interestingness scores; (3) be used during post-processing to select interesting patterns (Larose 2004). Fortunately, interestingness measures have been the focus of considerable research interest. All of the method groups outlined above have associated suggestions of interestingness measures, although most are objective.

12.2.5 Data Formats

An important aspect of working with game telemetry data is how they are stored and accessed. It is one thing having collected behavioral data from ten million players, another to store these in a way that makes it as easy as possible to apply data mining techniques to them. There are currently a plethora of database formats available, with SQL/MySQL being one of the most commonly used, and used to be the default for new web applications. However, SQL has problems with scaling up to very large datasets, despite recent innovations such as SSD enhancements and 32+ core scalability, and can be overly complex for many operations (and making changes to large databases can be hard). Therefore, in recent years more “elastic” means of data storage, running on cloud computing frameworks with up to 100’s of servers, offering scaling on demand. These new database formats are commonly referred to as “NoSQL” (and NewSQL) and have become popular in big data contexts due to the need for fast, efficient data access. A full review of different database formats is dramatically out of scope, but interested readers can find useful information in Chaps. 6 and 7 of this book. It is also recommended to look the NoSQL database formats MongoDB, Cassandra, Couch and HBase (Hadoop), for information on newer database formats. In general, the Net is a good source for information on database formats.

Data mining methods are applicable to any kind of data or media and independent on the specifics of the repository of the data (relational database, unstructured database, multimedia database, time-series database, flat file, object-oriented database, spatial database, etc.). However, algorithms may vary when applied to different types of data, e.g. images vs. behavior measures.

12.2.6 Tools for Game Data Mining

There are a wide variety of software tools available for data mining. Some are specialized for particular sectors of industry or research; others are more open and accommodating to game data mining. However, in our experience, some software vendors have a tendency to market analytical software as being “plug and play” applications that will provide solutions to all kinds of problems without the need for human interaction. This is blatantly not the case there is a strong need for a human element in data mining, possibly especially in games, where the fundamental goal of providing the user with a good experience is at the forefront; results, therefore, need to be interpreted with user experience in mind (Drachen et al. 2009).

In recent years, several companies have started to offer middleware technologies specifically for game data mining or game analytics (e.g. www.gameanalytics.com, www.playtomic.com, www.honeytracks.com, www.kontagent.com), supplementing the tools and services offered by traditional analytics companies. However, game-specific data mining tools remain in their infancy, and traditional game data mining companies, used to working with, e.g., business analysis or web analytics, do not always have the intimate understanding of game design necessary to fully understand game telemetry data, and deliver relevant and interesting results. This has led to several major publishers, notably Microsoft Studios Research, to develop their own solutions to game analytics (Kim et al. 2008). The barrier of entry for non-experts in game design and data mining remains, therefore, relatively high. However, the current rapid development in game telemetry analysis favors a wider availability of solutions and methods evolving over the next few years.

In the open-source market, there are many freely available tools developed by researchers and practitioners that are useful to novices and experts alike, for example tools like Weka (www.cs.waikato.ac.nz/ml/weka/), which is used to supervised learning, RapidMiner, a general data mining tool or Shogun, a library for large scale machine learning (www.shogun-toolbox.org), Pymf, a toolbox for matrix factorization in Python (pymf.googlecode.com), or QGIS for spatial problems (www.qgis.com). There are many of these tools (see e.g. Chaps. 7, 10, and 14). At the practical level, the easiest way to locate an open-source toolbox useful to a particular data mining task is to figure out what type of problem we are dealing with, and then browse the Net for relevant tools.

12.2.7 Practical Issues in Game Data Mining

There are a range of important issues to consider when planning to or performing collection of game telemetry and mining of this type of data. Confidentiality of user data, security of hosting servers, transparency of analysis results, and effective preprocessing approaches are among the most important. In this section, these issues and their implications are briefly introduced, but the interested reader is strongly advised to consult the literature at the end of the chapter for more detail.

Transparency: the patterns discovered by data mining tools are useful only if they are interesting and understandable to the user they are aimed for. Any data mining result (model) should be as transparent as possible, i.e. the result should describe a pattern that is intuitively interpretable and which is followed by an explanation, targeted at the specific stakeholder or user of the result e.g. decision trees are intuitive and almost self-explanatory in terms of their results, but neural networks are comparatively opaque to the non-expert (as are non-linear models in general). For example, a game designer may not be a statistics expert and therefore providing the results of a variance analysis in the standard statistical reporting form (a series of values), will not be conducive to the designer understanding the result and being able to act upon it. Transparency is vital to ensure that the various users of game data mining results are able to understand and act upon them. Another issue in visualizations is screen real-estate, information rendering and user-pattern interaction. Interacting with raw data or mining results is important, because it provides the means for users to focus and refine the mining tasks. Additionally, it allows users to model the discovered knowledge from different angles or conceptual levels.

Data cleaning: data analysis can only be as good as the data that is being analyzed, and most algorithms assume the data to be noise-free. This is an important assumption. Depending on the technical back end, game telemetry data may be more or less complete or saddled with different types of problems. Data cleaning (or cleansing) is the process of detecting and removing inconsistencies from data, towards improving and ensuring the quality of the data (Han et al. 2005; Larose 2004).

Quality problems in raw data come in many forms, e.g. misspellings during data entry, missing information or the presence of invalid data. When multiple sources of data are integrated, for example in a data warehouse, or analysis run across multiple data sources (e.g. telemetry from different games), the requirement for careful data cleaning increases due to the potential for error introduced when datasets are combined.

Performing data mining on low-quality data (“dirty data”), with, for example, missing or duplicate information, can compromise the validity and accuracy of the results, or even worse, can lead to outright wrong results, following the “garbage in, garbage out”-principle in data mining. As a consequence, data cleaning and data transformation (commonly referred to as pre-processing) is vital, but is often erroneously viewed as lost time. As frustrating as data cleaning may be, it is one of the most important phases of the knowledge discovery process. Data cleaning is a complex topic. Unfortunately, it is not possible to provide simple guidelines to address this topic. There is also a general lack of research in the area despite the importance.

Performance and sampling: many methods for data analysis and interpretation were not originally designed for the very large datasets that exist today. In game development, telemetry datasets easily reach the terabyte size for online social games or for large commercial games with hundreds of thousands or millions of players. In addition to the size of the data, the dimensionality of the data, i.e. the number of variables in the dataset (e.g. the number of variables such as completion time, class, level, etc., known for each player in a game), is decisive to the choice of data mining techniques. In general, the search space grows exponentially with the number of dimensions in a dataset, and its effect is so dramatic that it is currently one of the most important research problems in data mining.

Many techniques have issues with scalability and efficiency at large scales and dimensionalities, especially those that scale quadratically with dataset size, or algorithms with exponential or polynomial complexity (Mahlman et al. 2010). Sampling is a possible solution, i.e. mining part of the dataset rather than the whole, and extrapolating results from the sample to the whole dataset. Sampling has its own complexities and challenges, for example in relation to ensuring a representative sample that captures the features of the entire dataset. Sampling is covered in more detail in Chap. 9. Another approach is parallel programming, where the dataset is subdivided and results for each subset merged later.

Security: is an important issue with any game telemetry data collection, whether intended for low-level work or strategic decision making. Game telemetry data are generally considered confidential in the industry, and should be kept safe, which includes considerations on how to handle data access, transfer of data and transfer of results.

Social and privacy issues: One of the key issues in data mining is the question of individual privacy. The immense collections of data on people, and the many opportunities for collecting additional information, combined with data mining, makes it possible to analyze, e.g., routine business transactions, and obtain a substantial amount of information about the habits and preferences of individuals or businesses. Additionally, when data is collected for player profiling, behavior, correlations of personal data with other information, and so forth, sensitive and private information about individuals or businesses is collected and stored. This is controversial given the confidential nature of such data, and the potential illegal access to it. Another issue is how the data is being used. Because this type of data is valuable, databases of all kinds are traded. It is, thus, important to be aware of what data and analysis results that are being distributed, e.g. email addresses of players.

Collection strategies: There are two fundamental ways to obtain data from an installed game client or hardware unit (e.g. Xbox 360, PS3, PSP, smartphone), irrespective of the protocol employed (e.g. restAPI). Choosing the right strategy for capturing data from game clients is vital to avoid excessive data cleaning issues and data loss. There are pros and cons to both approaches, as follows (adopted from Mahlman et al. (2010)):

- **Fire and forget:** game telemetry data are stored locally in queues. Depending on the memory allocated to telemetry tracking, the size of the queue can vary. The game client will attempt to transmit data to the collection server, but may or may not receive confirmation of receipt from the server. If a queue is full, the oldest stored data are deleted first to make space for new data. This solution ensures a specific memory use and is, thus, useful for mobile platforms or consoles where memory resources are limited; or for high-frequency data (e.g. navigation), where some random losses are unimportant.
- **Reliable metrics:** the game client keeps storing telemetry data until they have been successfully transferred to the collection server, and confirmation of receipt has been received. The solution is resistant to loss and useful in situations, where the data must be collected as completely as possible, e.g., during playtests. In both

cases, a key rule is that game execution must not be affected by the collection or transfer of telemetry data to the collection server. The approaches can be combined, e.g., using limited queuing for navigational data and unlimited queuing for important variables.

12.2.8 Data Mining Approaches

It is difficult to generalize about data mining methods given the many fields of research and business that employ data mining techniques. However, the various methods are usually divided into either the categories **descriptive/prescriptive** or **unsupervised/supervised learning**. Depending on the person or book being consulted, either of these two divisions will be used – they are not; however, completely interchangeable – descriptive data mining is not the same thing as unsupervised learning, for example. To be more precise, predictive/descriptive data mining are concepts, and supervised/unsupervised learning are concrete categories of methods – and not the only ones used for data mining, although the main ones. This means that for example correlation methods are referred to as descriptive data mining, but not assigned to the unsupervised learning group of data mining methods. Similarly, interpolation is a technique used for prescriptive data mining, but can in at least some cases be argued to be not a form of supervised learning. As with so many other things, the difference is good to be aware of, but not vital in practice. In this chapter, we adopt the division of methods into unsupervised and supervised categories.

Descriptive data mining is used to describe the general properties of existing data in a concise way. In addition, it presents any interesting characteristics of the data without having a predefined target. For example, exploring the number of daily users and pointing to a sharp increase in active users on a specific day, say Saturdays. Some authors equate descriptive data mining with statistics.

Predictive data mining is used to forecast explicit valued, based on patterns determined from known data. In other words, it is used to attempt to predict something based on inference on the data at hand. For example, predict how many paying users a game will have based on data on previous subscriptions.

Supervised learning originates in machine learning – a branch of artificial intelligence science that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on data. A learning algorithm takes advantage of a test dataset, “training data” (observed examples), to capture characteristics of interest of the underlying, unknown probability distribution of data and make intelligent decisions based on their properties. In supervised learning, training data is combined with knowledge of desired outputs. The output of the algorithm can be a continuous value (regression) or a prediction of a class label of the input object (classification). The task of the supervised learning algorithm (the “learner”) is to predict the value of the function for any valid input, after seeing a number of

training examples (i.e. pair of input and target output). In order to achieve this ability, the algorithm has to generalize from the training data to unknown situations in a way that is reasonable. In the context of digital games, predictive data mining can be used to forecast when a player will stop playing, if a player will convert from a non-paying to a paying user, what types of items players will purchase, classify player behavior, etc.

Unsupervised learning also originates in machine learning, and also focuses on fitting a model to observations. However, unlike supervised learning, there is no a priori output. The input objects are generally treated as random variables, and a density model built for the dataset. For example, if we want to classify player behavior, we can use unsupervised learning if we not know how the behaviors varied, or if no previous classes had been defined. We can use supervised learning if, for example, we already run a classification on earlier data, and are interested in fitting some new players into these pre-defined classes.

12.2.9 Data Mining Methods

The classification of data mining methods beyond descriptive/predictive and supervised/unsupervised has always been a somewhat sensitive issue in data mining, leading to some confusion when attempting to learn about the different methods – a popular class or concept may have dozens of different names (Han et al. 2005). In the context of game data mining, some of the most common methods used are:

Description: is when analysts are simply trying to describe patterns of trends in game data, and is usually accomplished using Explorative Data Analysis (EDA), which is a graphical method for exploring data in search of trends or patterns. For example, plotting class level vs. playtime per level in a bar chart across six classes in a MMORPG, and finding that the “warrior” class progresses more slowly than the other classes. Descriptions of patterns often suggest possible explanations for them. EDA is particularly useful for basic analysis and for obtaining an understanding of the game data prior to the application of advanced algorithms.

Characterization: is simply the summation of general features of objects in a target class (or sample), producing a characteristic rule. For example, we may want to characterize all players who complete the first 100 quests in a RPG in less than 5 h (an example of characterization is shown in Fig. 12.2, where telemetry data from the ranges at which weapons were used in a FPS are averaged across the weapons).

Discrimination: is when features of objects across two classes (or samples) is compared. For example, comparing the most popular item purchases for players between 10–15 and 16–20 years, or comparing the navigation path of two types of players through a game level (Chaps. 7, 14, and 19; Drachen and Canossa 2011). Discrimination is identical to characterization except that discrimination results include comparative measures.

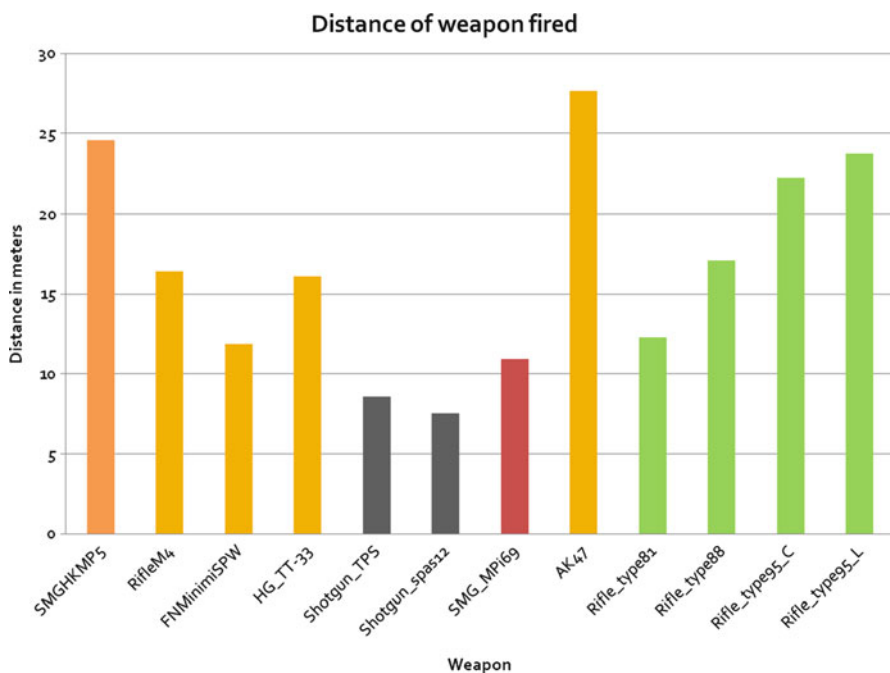


Fig. 12.2 An example of a descriptive analysis: A simple bar chart providing an overview of the average distance at which playtesters of the multiplayer shooter *Fragile Alliance* (Eidos Interactive, 2007) used different weapons, during early production of the game, for a particular map (note that the published version of the game has other rates and weaponry) (used with permission from IO Interactive)

Classification: is used to organize data into classes, which is hugely useful to game development. For example, classifying players based on their potential to become paying users vs. non-paying, or classifying player behavior in a shooter game to test if the players play the game as intended by the games' design. Classification uses class labels to order objects in a data collection, normally using a training data set where all objects are already associated with known class labels (e.g. playtime per level associated with character class). The classification algorithm used leans from the test data and builds a model that can be applied to other or future data.

Estimation: is similar to classification, but the target variable is numerical, not categorical. In statistics, methods such as regression and correlation are estimation methods. For example, we are interested in knowing a value, not obtain information about how our data groups into distinct classes. For example, estimating how much money a player will spend on in-game items, or how long a player will continue playing a specific game. In estimation, models are built using training data of complete records, which provide the value of the target variable as well as the predictors (causal variables). For new observations, estimates of the value of the target variable are made, based on the values of the predictors. For example, using simple regression to find the relationship between two variables, such as playtime and money spent on in-game items.

Prediction: is reminiscent of classification and estimation, but with prediction, we want to know about the future. The core idea is to use a large number of known values to predict possible future values. For example, how many players an MMORPG will have 3 months into the future, or when there will only be 1,000 active players left in a social casual game or how many players are needed to reach the critical threshold when player communities become self-sustaining. There are many approaches to prediction, from traditional statistical methods to more specialized knowledge discovery methods, such as neural networks, decision tree analysis, and k-nearest neighbor (Mahlman et al. 2010). Prediction is one of the most widely applied data mining methods in the analysis of data from multi-player and massively multi-player persistent games, where predicting the effect of design changes or the behavior of the player community, is important for revenue. Prediction can be used to forecast in many contexts around game development and -publishing.

Clustering: is a lot like classification, in that the aim is to order data into classes. However, the class labels are unknown and it is up to the clustering algorithm to discover what the classes are and evaluate their acceptability. The core goal of clustering algorithms is to group or segment objects (e.g. players, asset, items, games or any observation, case or record) in such a way that the similarity between objects in one group (cluster) is high (intra-cluster similarity), while between groups is dissimilar (intercluster similarity).

Association (affinity): when performing an association analysis, the goal is to find features (attributes) that “go together”, thus defining association rules in the data. An association rule specifies that if X, then Y, e.g., “*if players buy Stribed Trousers of Strength +3, they will also buy Girdle of Charisma +2.*” The association rule is accompanied by a measure of support, and of confidence. The support threshold identifies the frequency of the features occurring, and the confidence threshold defines the probability one appears when the other does. For example, it may be found that out of 1,000 players, 500 bought the *Stribed Trousers of Strength +3*, and of those 500, 250 bought a *Girdle of Charisma +2*. The association rule then specifies: “*if players buy Stribed Trousers of Strength +3, they will also buy Girdle of Charisma +2, with a support of 50.*”

There are many other methods that can be used for game data mining, such as outlier analysis (looking at the exceptions to normal behavior, which can be pretty useful in the analysis of player behavior, e.g. for locating gold farmers), evolution analysis, and deviation analysis (the investigation of time related data that changes as a factor of time). However, these are out of scope for this chapter. The reader is referred to the reference list for further information.

12.3 Unsupervised Methods

As discussed above, unlike supervised models, in unsupervised learning there is no *a priori* defined output, i.e. we are not trying to predict target values, but rather focus on the intrinsic structure of and relations in the data. In particular, the data

mining algorithms searches for patterns among all variables of a given dataset. A traditional example is clustering, e.g. for classifying player behavior, causes for game crashes, etc. In such examples, we are not typically sure how these behaviors vary or whether particular causes are more typical than others. In the following sections some basic mathematical properties are described in the interest of accuracy.

In this section, we concentrate on a few examples of the application of unsupervised models for analyzing game telemetry data. We will give an overview over a few common methods applicable for unsupervised data analysis in games. We will demonstrate the usefulness of recent data mining techniques in terms of acquiring interpretable data representations.

12.3.1 Clustering

In the context of customer behavior analysis in computer game development, cluster (and classification) analysis provides a means for reducing the dimensionality of a dataset in order to find the most important features, and locate patterns which are expressed in terms of user behavior as a function of these features, which can be acted upon to test and refine a game design (or specific parameters of a design) (see Drachen et al. 2012, for a more in-depth discussion). For example, figuring out how people play the game or identifying groups of players who display unwanted behavior. Clustering is thus a highly useful data mining method, containing a plethora of algorithms, the most commonly used being k-means (Golub and van Loan 1996).

There are however notable challenges (Drachen et al. 2012):

1. The potentially high dimensionality of behavioral data from games
2. There is sometimes a need to mix datatypes, e.g. binominal and categorical features which makes normalization challenging
3. Telemetry datasets can be noisy
4. Clustering generally require informed decisions as to the number of clusters to extract
5. The results have to be actionable. What this means is that it should be possible to relate the results to the design of the game in question, which entails converting results to a language understandable by the target stakeholder group (designers, marketing, management etc.).

Successful clustering of player behaviors in computer games requires that these challenges are addressed. Furthermore, it is important to note that the integration of knowledge of the design of the game being investigated is necessary to guide the process of selecting which behavioral variables (or features) to work with. Also, depending on the goals of the analysis, different clustering algorithms may be more or less applicable (Thureau and Drachen 2011), because the algorithms have different properties (this is discussed in further detail below).

To exemplify the process of clustering, we applied k-means clustering to a simple bivariate (playtime and character level) dataset derived from *World of Warcraft* (approx. 70,000 characters) Fig. 12.3 shows the result, indicating that there are

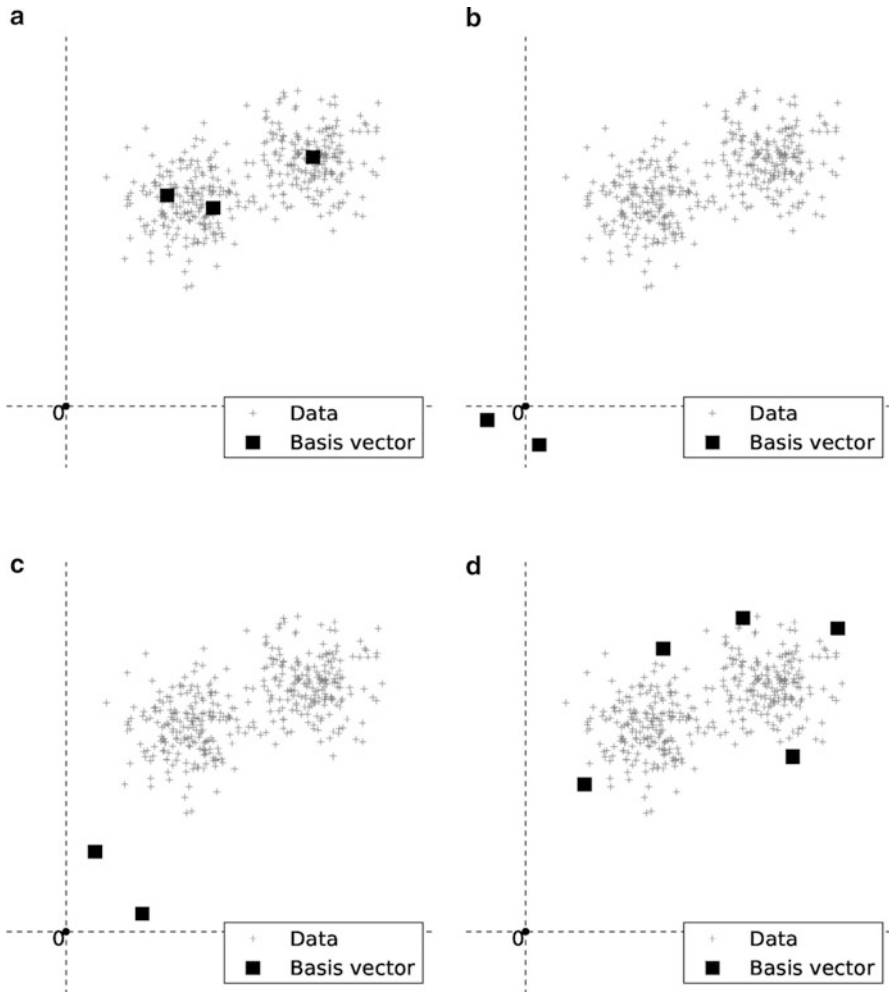


Fig. 12.3 Different cluster/matrix factorization methods can yield completely different views on the same game telemetry data. (a) k-means clustering, (b) PCA, (c) NMF, and (d) Archetypal analysis

roughly two or three separate “clouds” of behaviors in the dataset. It can be seen that the resulting cluster centroids (the central point in each cluster cloud) or basis vectors reside within the data and represent certain cluster regions (top left diagram), i.e. one particular cluster center can now be used to represent a vast number of data samples. Each data sample is assigned to exactly one cluster centroid. This is, arguably, the most common way of cluster analysis as it tries to approximate larger dense data distributions by one particular cluster centroid. However, various other unsupervised methods exist (e.g. Principal Component Analysis, Archetypal Analysis and Non-negative Matrix Factorization – results of which are shown in the

other diagrams of Fig. 12.3) in that yield different cluster centroid locations and are less restrictive with respect to cluster membership of individual data samples. Often, these methods have advantages over the standard mean-based approach and can lead to more interpretable data representation (Thurau and Drachen 2011; Drachen et al. 2012).

12.3.1.1 Clustering – Formal Basis

Mathematically, when running a cluster analysis we are dealing with n samples of d -dimensional vectorial data gathered in a data matrix $\mathbf{V}^{d \times n}$. The problem of determining useful clusters corresponds to finding a set of $k \ll n$ centroid vectors $\mathbf{W}^{d \times k}$ (note: not all clustering methods use centroid vectors, see e.g. AA below). If we express the membership of data points in \mathbf{V} to the centroids in \mathbf{W} using a coefficient matrix $\mathbf{H}^{k \times n}$, we note that clustering can be cast as a matrix factorization problem which aims at minimizing the expected Frobenius norm $\|\mathbf{V} - \mathbf{WH}\|$. For example, for k-means clustering, where each data sample exclusively belongs to a particular cluster center, the columns of \mathbf{H} are all zeros, except the row to the i -th cluster centroid which is 1, assuming the i -th cluster centroid is the closest.

Generalizing clustering as a matrix factorization task immediately extends the range of applicable approaches. Common methods to achieve the desired factorization include principal component analysis (PCA) (Jolliffe 1986; Golub and van Loan 1996), non-negative Matrix Factorization (NMF) (Paatero and Tapper 1994; Lee and Seung 1999), or Archetypal Analysis (AA) (Cutler and Breiman 1994), among others. However, resulting basis vectors (or cluster centroids) \mathbf{W} considerably differ among the mentioned algorithms. While all mentioned methods roughly try to minimize the same criterion (the expected norm $\|\mathbf{V} - \mathbf{WH}\|$), they impose different constraints that yield different matrix factors. For example, PCA (Fig. 12.3b) constrains \mathbf{W} to be composed of orthonormal vectors and produces a dense \mathbf{H} , k-means clustering constrains \mathbf{H} to unary vectors, and NMF (Fig. 12.3c) assumes \mathbf{V} , \mathbf{W} , and \mathbf{H} to be non-negative matrices and often leads to sparse representations of the data. While the mentioned factorizations have their specific application in data analysis, it is often not obvious which method to choose for a particular task. Therefore, we will first take a closer look at the specific requirements of data analysis in games.

A common goal of unsupervised data analysis in games is player categorization, or grouping (the supervised learning equivalent is classification), ideally resulting in representations of the telemetry data which is interpretable by non-experts. Ideally, one could assign a simple expressive label to each found basis vector or centroid. While there is no objective criterion on what a descriptive representation is, it is widely assumed that approaches yield interpretable results when they embed the data in lower dimensional spaces whose basis vectors \mathbf{W} correspond to actual data points. This is e.g. the case for Archetype Analysis (AA) as the basis vectors or archetypes the method produces are restricted to being sparse mixtures of individual data points. This makes the method interesting as a means for game data mining as it does not require expert knowledge to interpret the results. This contrasts with

other dimensionality reduction methods, such as PCA (Jolliffe 1986), where the resulting elements can lack physical meaning (Fig. 12.3), and NMF, which yields characteristic parts (Fig. 12.3) (Finesso and Spreij 2004). K-means clustering is similar to AA as the basis vectors reside within cluster regions of the data samples. However, the centroids do not necessarily have to reside on existing data samples.

Taking a closer look at Archetypal Analysis, we note that it uses a constraint that expresses data as convex combinations of certain points in \mathbf{V} , exemplary resulting clusters (Fig. 12.3). It can be seen that the resulting basis vectors come to reside on the convex hull of the data distribution, and thus, unlike most other methods, data is expressed by the most extreme and not the most average samples. Searching for certain extremal elements in a set of data as it is done for AA accommodates human cognition, since memorable insights and experiences typically occur in form of extremes rather than as averages (on a side note, Philosophers and Psychologists have noted this for long, since explanations of the world in terms of archetypes date back to Plato). In contrast, k-means clustering focuses on the average, and is therefore in the context of other centroids usually more difficult to interpret. While the centroid vectors all cover different regions of the data space, their overall similarity is often too high as it would help a human observer in assigning it a concrete label, i.e. description of the cluster.

The AA problem can be formulated as $\mathbf{V} \approx \mathbf{VGH}$ where $\mathbf{G} \in \mathbf{R}^{n \times k}$, $\mathbf{H} \in \mathbf{R}^{k \times n}$ are coefficient matrices such that \mathbf{H} is restricted to convexity and \mathbf{G} is restricted to unary column vectors $\mathbf{1}^T \mathbf{h}_j = 1, \mathbf{h}_j \mathbf{1} \geq \mathbf{0}$, and $\mathbf{g}_i = [\mathbf{0}, \dots, \mathbf{0}, \mathbf{1}, \mathbf{0}, \dots, \mathbf{0}]^T$. In other words, the factorization approximates \mathbf{V} using convex combinations where the basis vectors $\mathbf{W} = \mathbf{VG}$ are data points selected from \mathbf{V} . The goal now is to determine a basis that minimizes the Frobenius norm $E = \|\mathbf{V} - \mathbf{VGH}\|^2 = \|\mathbf{V} - \mathbf{WH}\|^2$.

When minimizing the Frobenius norm, we have to simultaneously optimize \mathbf{W} and \mathbf{H} , which is generally considered a difficult problem and known to suffer from many local minima. AA, as introduced in (Cutler and Breiman 1994), applies an alternating least squares procedure, where each iteration solves several constrained quadratic optimization problems. It solves the case where \mathbf{G} is restricted to convexity instead of to unarity. It is important to note that Archetypal Analysis originally was restricted to smaller datasets due to the demanding computation; very recent work has discovered ways of extending Archetypal Analysis to large-scale datasets (Thureau et al. 2009, 2010), making the method effective for implementation in the context of game metrics. Namely, the authors introduced convex-hull non-negative matrix factorization (CHNMF) and simplex-volume maximization as an approximation to AA (Thureau et al. 2009, 2010) (A Python implementation of the two methods is available from pymf.googlecode.com).

12.3.1.2 Example 1: Clustering Players in Battlefield 2: Bad Company 2

The following case study is drawn from Drachen et al. (2012), and is focused on

Battlefield 2: Bad Company 2 (BF2BC2) (2010, Electronic Arts), a first person shooter with tactical wargame elements, usually played in online multiplayer supporting up to 32 players, but including off-line (single-player campaign) capability.

In the multi-player mode of BF2BC2, each player controls one character in a team, playing against another team. There are various types of modes of play, and players can select between a range of classes, referred to in the game as “kits”. These are: Assault, Demolition, Specialist, Recon and Support. Each class provides different starting equipment. In addition, players can earn awards, ranks and special equipment.

Drachen et al. (2012) used behavior telemetry data from randomly selected 10,000 BF2BC2 players, all playing on PC. A total of 11 variables (features) were included in their analysis, with some of these being compound features. Given the hundreds of possible behavioral variables that can be tracked from players in BF2BC2, selecting these 11 required consideration. Drachen et al. (2009), working with data from *Tomb Raider: Underworld*, suggested that any initial and explorative cluster or classification analysis of player behavior should focus on behaviors related to the central mechanics of a game, and this principle was adopted, leading to a selection of features relating to character performance (score, skill level, accuracy etc.) and game asset use (kit stats, vehicle use), and playtime – as follows (quoted from Drachen et al. 2012):

- **Score:** Total number of points scored
- **Skill level:** An aggregate measure of player skill
- **Total playtime:** The sum total of time the player’s account has been active
- **Kill/Death ratio:** K/D ratio, the number of kills the player has scores divided with the number of deaths suffered
- **Accuracy:** The percentage of hits scores with weapons
- **Score per minute:** The average number of points scored per minute of play while on active combat missions
- **Deaths per minute/Kills per minute:** Dpm/Kpm – Average deaths or kills per minute
- **Rounds played:** The number of game rounds the player has played
- **Kit stats:** The number of points scored with each kit (class) and the number of kills and deaths for each class
- **Vehicle use:** Total time spent in air, water, land-based or stationary vehicles

Following pre-processing and normalization of the telemetry data, two algorithms were applied to the data: k-means, which produce cluster centroids (Fig. 12.3), and Simplex Volume Maximization (SIVM), a variant of Archetype Analysis extended to large-scale datasets. SIVM does not look for commonalities between players, but rather archetypical (extreme) profiles that do not reside in dense cluster regions, but at the edges of the space spanned by the data points (Fig. 12.3). Both algorithms resulted in seven clusters, but the behavioral profiles that could be extracted from these varied somewhat – this is to be expected given the different natures of the algorithms. This number was decided upon using Scree plots and means squared error, two techniques for deciding on the number of clusters to work with. We will here focus on the results from the SIVM analysis, which resulted in the following behavioral profiles, three of which are largely independent of the classes in the game, and four which are closely related to them:

- **Assassins:** characterized by having extremely high Kill/Death ratios and highest Kpm ratio, but surprisingly low-middle playtime. Assassins are the most lethal players in the game, but also highly specialized.
- **Veterans:** are the all-round elite. Where the Assassins are specialized, the Veterans display the highest or second highest values across all the behavioral variables measured, but have also invested a lot of playtime into the game, indicating that these players are committed and stable. They represent a small fraction of the players, however, on the scale of 2–4%.
- **Target dummies:** These are the opposite of the Veterans, with lowest or very low values for all the behavioral variables, comprising about a quarter of the players in the sample. They have not played BF2BC2 for long, have low K/D ratios, often get killed, and their Score per minute is the lowest of all the profiles. Their only redeeming factor is a middling Accuracy.
- **Assault-Recon:** These players display high performance with the Assault and Recon kits, correlating with high kill rates and death rates (they are on the front-line), and the second highest K/D rate overall. They also exhibit low accuracy, which may relate to the rapid-fire weapons associated with the assault class. Only about 1.5% of the players are included in this cluster.
- **Medic-Engineer:** These players have very high skill levels and accuracy, score many points (second only to Veterans) and drive in vehicles a lot. Only about 1% of the players are included in this cluster, representing a highly specialized type of behavior.
- **Assault “specialist”:** While this cluster of players mainly plays the assault class, they do it relatively badly. They die a lot, but have invested a lot of playtime into the game, with low skill, K/D ratio and accuracy. They are not quite at the level of the Target Dummies, but perhaps represent the typical novice player. About 5% of the players fall into this cluster.
- **Driver Engineers:** These players favor the Engineer class and have extremely high vehicle times (4 times higher than any other cluster), i.e. they spend a lot of time driving, sailing or flying the various kinds of vehicles in BF2BC2. They have high playtimes, scores and accuracy, very high K/D ratio but kill very few players, and also die rarely. Only about 1% of the players are included in this cluster.

The latter four behavioral profiles represent well two of the fundamental ways of playing *BF2BC2*, either combat-oriented or support-oriented.

12.3.1.3 Example 2: Comparing Clustering Algorithms in *World of Warcraft*

Our intention here is to demonstrate how common clustering techniques perform on game metric data with respect to (a) descriptive representations, and (b) cluster separation. Four different clustering algorithms are applied to find clusters in this dataset, the results compared and evaluated, and recommendations made. The example presented here is drawn from Thureau and Drachen (2011).

The data for this case study contains a selection of approximately 70,000 player records, covering a period of about 5 years. The telemetry data are player/guild logs gathered from *WarCraft Realms* (<http://www.warcraftrealms.com>). The logs show for a certain number of dates the records of currently online players from European and United States *World of Warcraft* realms. In addition, character names, level, class, and guild membership are recorded.

The *World of Warcraft* dataset contains a set of players recordings, their online time, and their level for a specific date. We aggregate the recordings into a 2:555 dimensional feature vector, where each entry corresponds to the level the player reached for each day in the last 6 years. Note that the maximal level of a character was increased twice via expansion packs (from levels 60 to 70 and 70 to 80) during the period of recording (and in December 7th 2010 a third time, following the end of the data logging period, from levels 80 to 85), usually when a new expansion got released. We applied AA, NMF, k-means, and PCA to the dataset. Note that unsupervised methods usually suffer from the problem of having no objective way of defining threshold values, which makes the definition of the number of classes (or cluster centroids) to use a subjective decision. These aspects of classification analysis add to the difficulty in adopting these methods by non-experts in a game design/development context. For the presented experiments we set the number of basis vectors/classes to $k=8$ (note that we only visualized the first five) based on a consideration of variance explained vs. retaining a useful number of basis vectors with respect to the end goal being to produce player classes that are significantly different behaviorally.

The resulting basis vectors or cluster centroids for AA are visualized in Fig. 12.4, and for PCA, K-means and NMF in Figs. 12.5, 12.6, and 12.7, respectively. For AA, for example, the left most plot shows the level/time history plot of a specific player who only very slowly increased his experience level from level 10 to level 20, and Fig. 12.4 (second plot from the left) shows a player who quickly increased his level to 70, and then after some time to level 80. These two player types can be immediately labeled as: “casual player” and “hardcore player”. Comparing the resulting basis vectors of the different methods shows that only for k-means clustering and AA we obtained an interpretable factorization. However, the k-means centroids (Fig. 12.5) are overall very similar and do not allow a straight-forward labeling. Basically, they all show is the same curve where only the slopes vary slightly. In contrast, the AA basis vectors in Fig. 12.4 are intuitively easier to interpret. From these, we can also make assumptions about the leveling behavior of the players. The steepest increase in the level seems to correlate with the release of expansion packs and the simultaneous increase of the maximal level. The basis vectors of PCA and NMF are, as expected given the nature of the algorithms, not or only partly interpretable. However, this does not necessarily mean they are useless. We could think about various tasks were a representation of individual by meaningful parts (NMF) is desired. For example, it is reasonable to assume that social groups (guilds in *World of Warcraft*) consist of linear non-negative combinations of meaningful parts, e.g. leaders and followers. This could be captured more accurately using NMF, as it does not restrict the basis vectors to actually existing data samples.

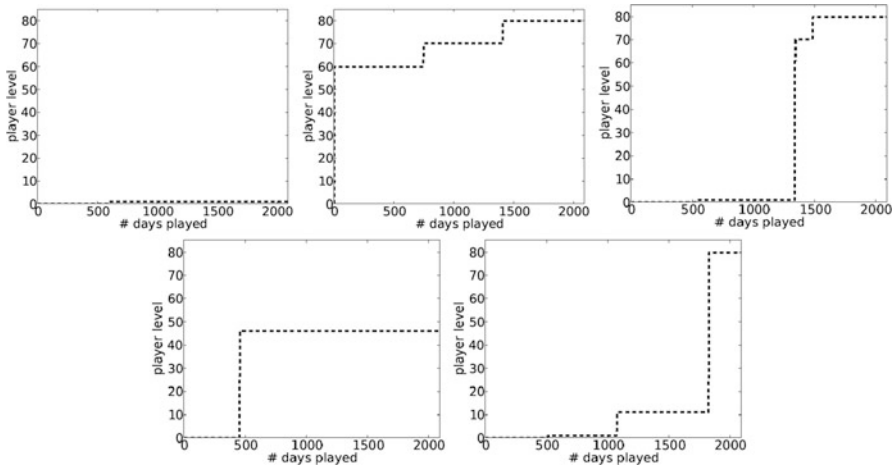


Fig. 12.4 Basis vectors for Archetypal Analysis. These reside on data samples (players in this case). All basis vectors correspond to legal player behavior (e.g. players do not loose levels). Note the straight line segments which map directly to level increases

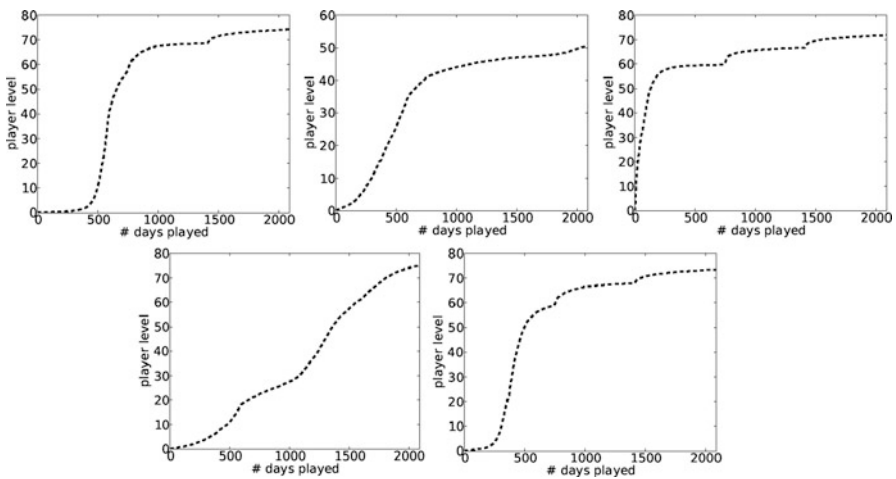


Fig. 12.5 Cluster centroids for k-means clustering reside on center locations of cluster regions. While they accurately represent a broad number of players, they are overall very similar to each other and do not allow straight forward interpretation

Besides a descriptive representation a quantitative discrimination of player types is desirable, i.e. how many players that belong to each behavioral class. This, however, is only fully supported using k-means clustering as it is the only method that builds hard cluster assignments, with each sample belonging to only one particular cluster. The other methods are usually soft (or more precisely linear, convex, or non-negative) combinations of their basis vectors. This means that players are expressed in terms of

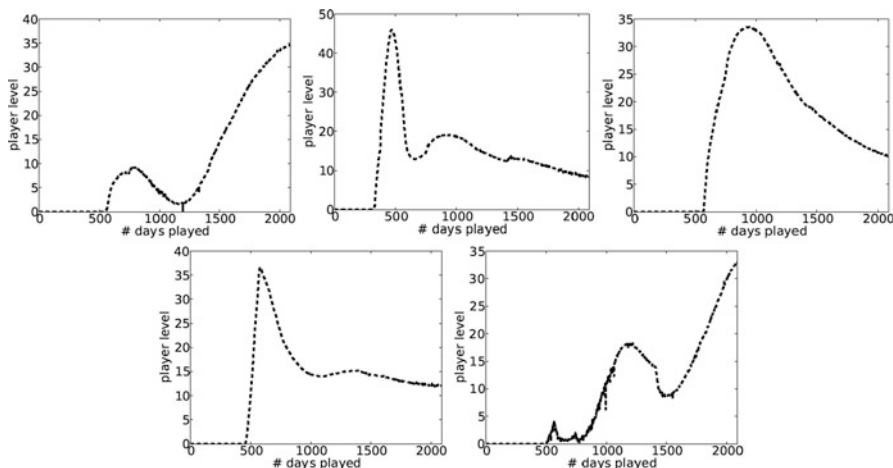


Fig. 12.6 Basis vectors for non-negative matrix factorization represent parts of original data samples. As they are strictly positive, they allow for interpretation but do not in this case correspond to actually existing players or behaviors that are possible in the game, e.g. characters are seen to loose levels

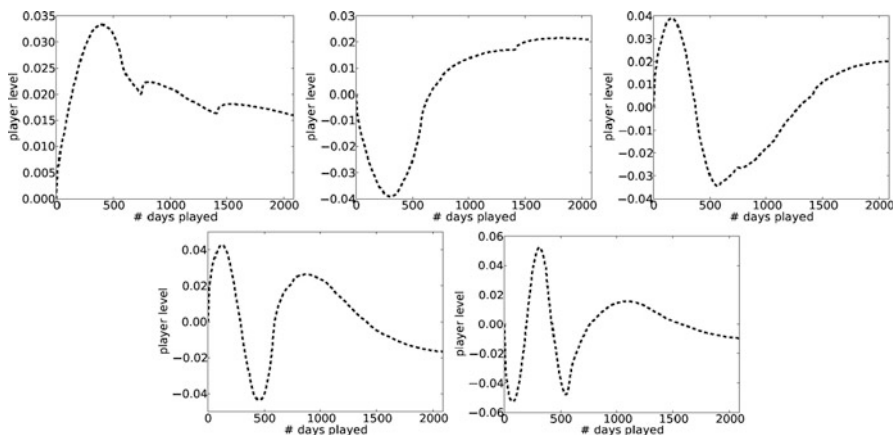


Fig. 12.7 Basis vectors for principal component analysis. These do not correspond to actual players, and correspond to behaviors that are not possible in the game, e.g. loss of character levels

their relationship to each of the eight behavioral profiles (basis vectors) located, and summarily grouped (clustered) according to their distribution in the space spanned by the basis vectors. For the numbers of players belonging to each basis vector provided here, players have been assigned to the nearest basis vector (behavioral profile). This provides clear profile divisions; however, a more precise way of grouping players is to define clusters in the space extended by the eight basis vectors. The results indicate that the distribution of players to eight basis vectors across the four methods included are not similar, with AA and PCA indicating three large groups, and k-means and NMF a division into four large and four smaller groups each (Fig. 12.8).

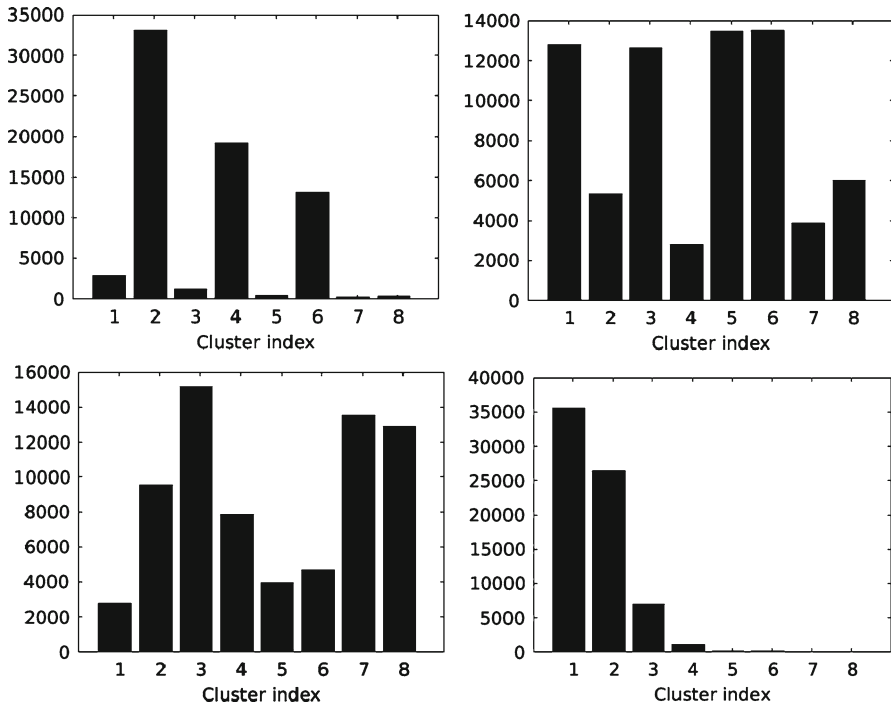


Fig. 12.8 Hard assignment of data samples to cluster centroids for (from top left and clockwise) AA, k-means, PCA and NMF. The bar charts highlight that the solutions generated by the four algorithms varies

12.3.1.4 The Evolution of Social Groups in *World of Warcraft*

Extracting meaningful information from very large amounts of data is a non-trivial task. Especially, if it is not entirely clear what to look out for. In these situations data mining resembles the proverbial search for a needle in a haystack.

Thurau and Bauckhage (2010) proposed the use of Convex-Hull Non-Negative Matrix Factorization (CH-NMF) as an efficient approach towards AA-like data embeddings by means of constrained matrix factorization. The goal was to try to get an accessible and interpretable description of very large amounts of game telemetry data, this towards developing an analysis of the development of guilds over time.

The dataset used by Thurau and Bauckhage (2010) in this example is similar to the one used in the above example (Section 12.3.1.3), but consists of 192 million recordings of 18 million characters belonging to 1.4 million guilds, and cover a period of 4 years, starting in 2005 (when *World of Warcraft* was released) and ending in early 2009.

The data recorded (roughly) summarizes some of the social in-game activities of players. That is to say, we know when players joined or left a guild, how many players were with a guild at what time, and how character experience levels were distributed among the members of a guild, as well as information about class and race. As mentioned before, the player's experience level provides a measure of skill for a

particular player. While a guild with a large number of high level players is more likely to be successful, a guild of only low level players is basically excluded from a large amount of the game content.

The distribution of experience levels among guilds, i.e. the number of players of a certain level that are with a particular guild, therefore provides a feature that characterizes a guild in terms of game success. The distribution can be approximated by means of building a histogram over experience levels of guild members (e.g. eight bins of ten levels each). If we build these histograms over all observations of a specific period of time, they also summarize the temporal evolution of a guild. A brief example should clarify this: If a guild is newly formed by level 80 players, it does not contain any observations of level 10 players and the corresponding histogram bin will be empty. A guild which is formed by level 10 players should, over a longer period, also have observations of level 40, 60, 80 (and intermediate levels) players, as the guild members usually increase their level over time.

In order to obtain an interpretable categorization of, Thureau and Bauckhage (2010) applied Archetype Analysis (more precisely its large-scale variant CH-NMF (Thureau et al. 2009)) to 1.4 million such guild histograms, containing data covering a period of 4 years. The result suggested eight clusters (basis vectors). A number of different basis vector numbers were tested, but it was found that eight basis vectors provide a convenient tradeoff between granularity and convenient visualization.

Following the definitions of CH-NMF, each basis vector resides on the convex hull (the outer surface of the point cloud in multi-dimensional space) of all the individual guild histograms, and thereby each basis vector represents an “archetypal” guild.

As noted before, this makes the basis vectors easy to interpret as there is usually only one salient characteristic – e.g. a guild comprised only of level 80 characters, or where the players level very rapidly from 0 to 60. Figure 12.9 shows a (constructed – due to copyright rules the original illustrations could not be presented here) example of a cluster centroid, i.e. an archetypal histogram. In the current case, the archetypal guilds are distinguishable from each other:

The eight basis vectors describe the following types of overall guild behavior:

1. Formed early, then disbanded
2. Active till the 2nd game expansion (*Wrath of the Lich King*), then disbanded
3. Seldom active
4. Formed before 2nd game update, then very active
5. Increasing activity, then disbanded
6. Increasing activity till the first expansion (*The Burning Crusade*), then disbanded
7. Active for character levels 10–80
8. Active between the 1st and 2nd expansion only

A wide variety of guilds are formed by a convex combination of these archetypal guilds, e.g. a guild can exhibit traits of both guild 2 and 5 for example.

Running the same data through k-means, Thureau and Bauckhage (2010) found that many of the resulting basis vectors were not as readily interpretable as the CH-NMF results, and the basis vectors tending towards being similar (an effect of the distribution of the data points in the variability space as well as the centroid-seeking behavior of the k-means algorithm).

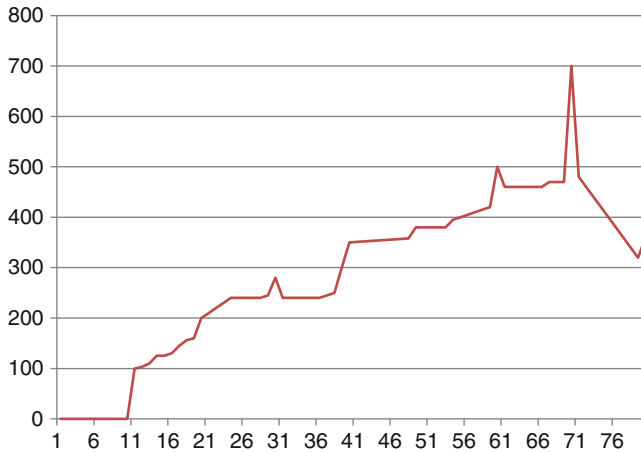


Fig. 12.9 Example of a basis vector resulting from the application of CH-NMF to the *World of Warcraft* guild dataset (constructed example). The x-axis denotes the level histogram bin, the y-axis denotes the number of observations for this bin. The guild described here has a gradually increasing number of players in the lower levels, with a noticeable spike and plateau structure at level 60–70, and a major spike at level 70

In order to obtain a better understanding of the development of the guilds over time, Thureau and Bauckhage (2010) projected time slices (90 days, 180 days, 1 year, 2 years, 3 years, 4 years) of the guilds into the space spanned by the CH-NMF basis vectors. They noticed that the total number of guilds (including disbanded guilds) increased considerably over time, following a roughly exponential growth rate, but also with a high abandonment rate. Also, a huge part of the guild space is densely covered – most guilds fall into the category of seldom active guilds (this could also indicate very small guilds) or are close to it. There is only a small number of guilds (still, many thousands) that completely fall into other categories (the eight basis vectors mentioned above).

On a final note, Thureau and Bauckhage (2010) did not find any significant differences between the development of guilds on US and EU servers.

12.3.2 *Player Classification in Tomb Raider: Underworld*

A Self-Organizing Map (or SOM) is a form of artificial neural network that is used in unsupervised learning to look for low-dimensional representations of the input data, similar to multidimensional scaling (Summit Kohonen 2001). For example, to find the main ways in which a group of people play a game. The input data are in this case gameplay metrics, the output are the classes into which the players are collected, along with the properties of the classes. For example, one class might be characterized by completing the game really fast, another completing the game really slow, and the third by not completing it at all.

SOMs (also called Kohonen maps after the inventor) work like most neural networks, in that the first step is building a model based on a training dataset, subsequently applying the model to map the main dataset. For example, in a 10,000 player sample, 1,000 could be used as training data, and then the SOM model achieved applied to the remaining 9,000. Without going into details, an SOM consists of neurons organized in a low-dimensional grid (usually two or three dimensions only). According to Drachen and Canossa (2009): “Each neuron on the grid (map) is connected to the input vector via a d -dimensional connection weight vector, $\mathbf{m} = \mathbf{m1}, \mathbf{m2}, \mathbf{md}$, where d is the size of the input vector, x . The connection weight vector is also named prototype or codebook vector. In addition to the input vector, the neurons are connected to neighbor neurons of the map through neighborhood interconnections which generate the structure of the map: rectangular and hexagonal lattices organized in two-dimensional sheet or three-dimensional toroid shapes are some of the most popular topologies used.” Please see (Summit Kohonen 2001) for a more detailed description.

Drachen and Canossa (2009) provide an example of how to field SOMs in practice. They used gameplay metrics data from 1,365 players of *Tomb Raider: Underworld*, including data on completion time, number of deaths, causes of death, etc. An SOM was used to find the emergent structures in the data, i.e. to classify the players into distinct groups based on their behavior. The analysis revealed four distinct classes of behavior, encompassing 93.54% of the player sample:

Cluster 1 (Veterans): (8.68%) characterized by having very few death events, and these mainly caused by the environment. Fast completion times. Generally perform very well in the game.

Cluster 2 (Solvers): (22.12%) die rarely, and very rarely use the help system in TRU, apparently preferring to solve the many puzzles in the game themselves. Take a long time to complete the game, indicating a slow-moving, careful style of play.

Cluster 3: (Pacifists): (46.18%) form the largest group of players, characterized by dying primarily from enemies. Completion time relatively fast and help requests minimal indicating some skill at playing the game in terms of navigation, but not a lot of experience with the shooter-elements of *Tomb Raider: Underworld* (the game used shooting substantially more than previous iterations of the Tomb Raider series).

Cluster 4: (Runners): (16.56%) die often and by enemies as well as the environment, use the help system fairly often but complete the game very fast.

The results showcase how SOMs are useful to evaluate game designs. In this case, the analysis indicates that players of the game utilize the affordances provided by the game, rather than simply adopting a specific strategy to complete the game. When evaluating if people play a game as intended by the design, the type of results generated by SOMs are immediately useful. However, the results of an SOM analysis are not intuitively understandable, and need to be translated into language that the intended user of the analysis can act upon. Finally, SOMs provide a good first-strike method for classifying player behavior, providing an overall view useful in guiding drill-down analysis.

12.3.3 Frequent Pattern Mining

Frequent pattern mining is the name used for a set of problems and techniques related to finding patterns and structures in data. While related to other unsupervised learning problems and techniques, such as clustering, frequent pattern mining differs both in the methods and in the format of input and output data — in particular, the latter is discrete and in the form of sets or strings. Several important problems in game data mining, e.g. player type identification and identification of common player behavior patterns, can be cast as frequent pattern mining problems of some form. The whole field of frequent pattern mining is less than two decades old (introduced in Agrawal et al. (1993)), yet several efficient algorithms exist for solving these problems.

Two particular types of frequent pattern mining problems that we will discuss here are *frequent itemset mining* and *frequent sequence mining*. Frequent itemset mining aims to find structure among data points that have no internal order, similar to most other data mining algorithms, whereas frequent sequence mining aims to find structure among data that has an inherent sequential (e.g. temporal) order. In the two sections below, we describe the problems, some main algorithms and applications for game data mining.

12.3.3.1 Frequent Itemset and Association Rule Mining

In frequent itemset mining, the base data takes the form of sets of instances (also called transactions) that each has a number of features (also called items). For example, a dataset of the items players bought in a social online game might contain five transactions as follows:

1. {*Sword of Grungni*, *Shirt of Awesomeness*, *Pretty Pet*}
2. {*Shirt of Awesomeness*, *Pretty Pet*, *Healing Potion*}
3. {*Sword of Grungni*, *Healing Potion*}
4. {*Shirt of Awesomeness*, *Sword of Grungni*, *Fancy Hat*, *Pretty Pet*}

The task for the frequent itemset mining algorithm is then to find all common sets of items, defined as those itemsets that have at least a minimum support (exists at least a minimum amount of times). If the support is set to 3, the following 1-itemsets (sets of only one item) can be found in the dataset described above: {*Sword of Grungni*}, {*Shirt of Awesomeness*} and {*Pretty Pet*}.

It is also possible to find one 2-itemset: {*Shirt of Awesomeness*, *Pretty Pet*}, as three of the transactions contain both *Shirt of Awesomeness* and *Pretty Pet*. Other itemsets of the same lengths are considered non-frequent as they recur less than three times. The original algorithm for mining frequent itemsets, which was published in 1993 and is still frequently used, is *Apriori* Agrawal et al. (1993). This algorithm functions by first scanning the database to find all frequent 1-itemsets, then proceeding to find all frequent 2-itemsets, then 3-itemsets etc. At each iteration, candidate itemsets of length n are generated by joining frequent itemsets of length $n - 1$; the frequency of each candidate itemset is evaluated before being added to the set of frequent itemsets. However, there exist several alternatives to this

algorithm. A prominent such alternative is the *FP-growth* algorithm, which finds frequent itemsets through building prefix trees Han et al. (2000).

Once a set of frequent itemsets has been found, association rules can be generated. Association rules are of the form $A \rightarrow B$, and could be read as “A implies B”. Each association rule has *support* (how common the precondition is in the dataset), *confidence* (how often the precondition leads to the consequence in the dataset) and *lift* (how much more common the consequence is in instances covered by the rule compared to the whole dataset). From the dataset and frequent itemsets above, the association rule *Shirt of Awesomeness* \rightarrow *Pretty Pet* can be derived with support 3 and confidence 1, whereas the rule *Shirt of Awesomeness* \rightarrow *Pretty Pet and Sword of Grungni* only has a confidence of only 1/3 and so would most likely not be selected as a useful association rule.

Frequent itemset mining can be used in several different ways for understanding game data. One way is to find patterns among players. If a database is organized so that each instance describes a separate player and the (binary or ordinal) attributes of each instance describe the player’s playing style (e.g. {violent, speedrunner, cleared_level_3, dies_from_falling}), frequent itemset mining can be used to find playing style characteristics that frequently co-occur.

12.3.3.2 Frequent Sequence Mining

Unlike frequent itemset mining, frequent sequence mining cannot be applied to separate, unordered instances (such as where each instance represents a player). Instead, frequent sequence mining requires the instances to be ordered in one or several sequences. The probably most common type of sequence data is temporal sequence data, where each instance represents the state of the system at some time t ; the interval between each instance might or might not be constant (in some terminology, an instance with all its features is called a symbol; identical instances map to the same symbol). The sequence mining problem is to, given a sequence or a set of sequences, find frequently occurring subsequences. For example, if the support threshold is set to 3, the sequence “abbabbcbdbb” has the frequent 3-sequence “abb” and the frequent 2-sequences “ab” and “bb”.

One of the most commonly used frequent sequence mining algorithms is SPADE Zaki (2001). SPADE works in a similar way to Apriori: first find frequent sequences of length 1 (i.e. single symbols), then combine these frequent sequences into candidate sequences of length 2, evaluate their frequency, combine into sequences of length 3 etc.

By virtue of being discrete-time systems, computer games constantly generate large amounts of sequential data. At one extreme, you could consider the complete state of a game at every frame (where a modern game usually runs at 30 or 60 frames per second) as a data stream to be mined. Of course, this data stream would generate far too much data for any existing algorithm to handle, and all practical applications require that only a few interesting features are logged rather than the complete game state. Additionally, often the temporal resolution is decreased. Identifying which features are interesting to log depends on the purpose of the data mining, but they may

include any aspect of the game state which is directly or indirectly affected by the player's actions, such as button presses, player character position and actions, non-player character position and actions, changing level geometry etcetera.

Kastbjerg (2011) combined frequent sequence mining with clustering in order to visualize the spatial form of common sequences of player actions in the multiplayer game *Heroes of Newerth* (2010, S2 Games). This work was an attempt to improve on the "heatmaps" that are commonly used to analyze player's movements in game levels, but which do not convey information on what players did at any particular point in time (see Chap. 17 for more on heatmaps).

A large, publicly available database of *Heroes of Newerth* game replays was mined; a few hundred thousand player traces were used in initial experiments (39,390 games, roughly 59 million events across 20,000 h of play data, average playtime per game around 30 min). For each game the actions taken by each player was recorded, along with the (in-game) time and position of the action. The most frequent 3-sequences of actions were then found using SPADE (the process is shown in Fig. 12.10). Once a particular 3-sequence had been decided on, the starting points of that sequence are clustered (for very frequent sequences, more than a hundred thousand repetitions of that sequences could be found in the database for a particular map). The user can then select a particular starting point cluster, and from there investigate how players typically move as they perform the chosen action sequence starting from the chosen point. This analysis revealed for example that the initial phase of a particular attack spell was frequently used before a teleport spell, and then unleashed on another part of the map; the spatial analysis pointed out which particular areas of the map this sequence typically started from and ended in (Fig. 12.10).

Sequences are not necessarily temporal data. Shaker et al. (2011) used frequent sequence mining as a way to find features with which to classify levels of the platform game *Super Mario Bros* (2004, Nintendo). The task was to classify which levels would be preferred over others, based on the survey results from over 700 players who had played at least two levels each. Here, the sequences are not based on the players' actions over time, but simply on scanning the levels from left to right. In the version of *Super Mario Bros* that was used for the experiments, levels are linear (the level starts at the left end and is won by reaching the right end) and constructed of blocks. A level is about 15 blocks high and a couple of 100 blocks long. Each level was turned into a single sequence with the same length as the level (one symbol per block). A few different ways were investigated for transforming each vertical slice of the level into a symbol, for example by simply using the height of the level at that point, or by basing the symbol on the topmost block in that slice. In the next step, SPADE was applied to the sequences generated from levels, in order to find commonly occurring subsequences, i.e. commonly used level segments — these included flat parts without any gaps, lines of coins, short gaps surrounded by platforms etc. Each level could then be categorized according to the incidence of these segments, and the segment counts were successfully used to form features when using a supervised learning algorithm to predict whether a particular level was preferred over another.

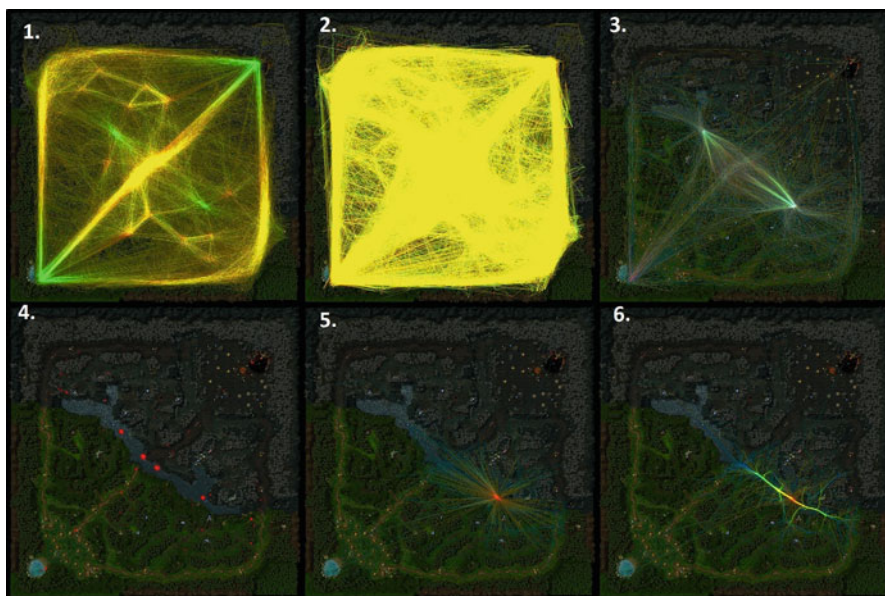


Fig. 12.10 Visualization of the sequence mining process applied to player traces in the *Heroes of Newerth* dataset. Image 1 show an intermediate step of the data loading and frequent sequence mining process SPADE. Each point on the map is colored in a heat map style, where the color of crossing edges is blended and increased in saturation. Image 2 shows the final result of the process. The result looks chaotic and is included to display how the data quickly becomes too excessive, thus the need for information extraction in the following steps: Image 3 shows a particular 4-sequence selected from the result pool of the previous sequence mining process. Image 4 shows the result of applying a modified version of FDEB on the start points of each instance in the sequence. In short the modified version omits the edge subdivision and intra attraction part (see Kastbjerg 2011, section 4.5 for an in-depth explanation). Image 5 shows the same as image 3, expect only sequences that start from a specific area, within a user defined radius. The particular area is selected by the user, but guide by the information found in step 4. Step 6 shows the final result, after the original FDEB algorithm has been applied to the edges selected in step 5 (Reproduced from Kastbjerg 2011 with permission)

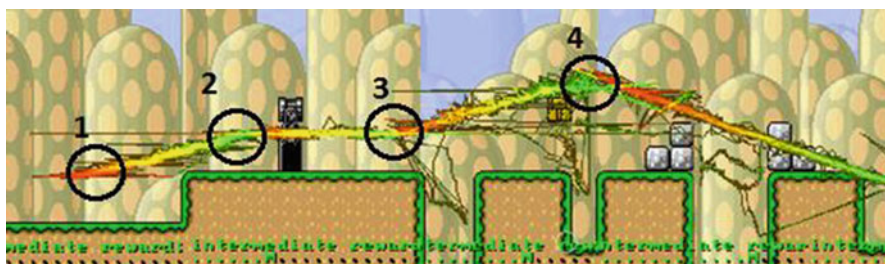


Fig. 12.11 A visualization of a frequent action sequence “jump-jump” from *Super Mario Bros.* Each circle marks a point where the player jumps (Reproduced from Kastbjerg 2011 with permission)

12.4 Supervised Learning

Supervised learning methods for data mining are drawn from Machine Learning (ML), a branch of artificial intelligence science that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on data, notably with the purpose of prediction. Machine learning – and thus supervised learning works from supervised training data. The training data, or signal, contains examples that the algorithms learn from, in order to be able to find the patterns or signals in data where the connection between input and output objects is unknown, e.g., which class to put a player in a given specific behavior. Supervised learning thus relies on a training dataset, or signal. The source of the signal defines the different clusters of all ML algorithms available. While in unsupervised learning the pattern (signal) is hidden in the internal structure of the data (interconnections among data attributes) and in reinforcement learning Sutton and Barto (1998) the training signal is derived as a reward from the learning environment, in supervised learning the signal is given in form of target data observations. Supervised learning is the process of training a function that approximates the mapping between attributes of the observations and the target observation. As a popular example for supervised learning, consider a machine being asked to distinguish between apples and pears (classes), given the color and size of the fruit (data attributes). Initially, the machine learner is trained on a number of attribute-class pair observations (i.e. training data providing the color and size of a number of apples and pears), from which it learns how to classify apples and pears. It can then subsequently be used to classify pears and apples based on the color and size input data only.

Popular supervised learning techniques include artificial neural networks, decision tree learning, support vector machines and bayesian learning (Bishop 2006). The primary use of supervised learning within games has been so far for the imitation of player behavior, the analysis of player behavior in online games, prediction of player behavior on massively multi-player online games, and notably for analysis of player behavior towards driving revenue in social online games (King and Chen 2009). For instance, the Drivatar system in *Forza Motorsport* (2005–2012, Microsoft Game Studios) is an artificial neural network that imitates the way a player drives a car and generates a race path that simulates the player's driving style. Similarly, the AI behind the player's deity avatar in *Black and White* (Electronic Arts 2003) uses supervised learning to imitate and respond to the player's actions and motivations.

A particular area of supervised learning in games is to imitate human playing behavior. Given a sufficiently large set of behavioral metrics data derived from players, supervised learning can be used for both imitating human playing behavior but also for predicting various aspects of the behavior. In AI research and industry, the main purpose of imitation is the creation of believable, human-like, non-player characters or similar computer-controlled entities, but there are also other purposes. Prediction can give answers to questions such as: “when will this player stop playing the game?” and: “how many times will this player use one weapon over another?” Essentially, supervised learning can be used for the prediction of any player attribute

given in the dataset. These kinds of questions are important to get answered during the development and testing of a game, and even after release. Supervised learning methods can thus be used to e.g. test and adjust designs, or even form the basis for systems controlling real-time adjustment of game features during play (e.g. dynamic difficulty adjustment).

12.4.1 Prediction Analysis and Decision Trees in Tomb Raider: Underworld

Prediction in data mining is performed with the goal of identifying a model of set of models that can be used to predict responses. For example, predicting which players will convert from non-paying to paying, or when particular players will stop playing a game. Notably in the context of social online games, prediction of player behavior, and design responses to changes in behavior, is important to ensure revenue. Prediction is similar to classification in that a model is constructed based on known data, and the model is used to predict unknown or missing values, e.g. future player behavior. The major method in prediction is regression, which generally attempts to construct either linear or non-linear models. Combining predictions from multiple models, which is particularly useful when the types of models included in an investigation are very different, is referred to as “stacking” or “stacked generalization”. Stacking is interesting because experiences have shown that predictions from multiple methods can yield more accurate predictions than any single method. When stacking, the predictions from different models are used as input to a meta-learner, which basically tries to combine the prediction models to create a “super-model” with the best predictions possible. The meta-learner can for example be neural network, which attempts to learn from the data how to combine the models for maximal accuracy in the predictions. Alternative approaches to combining prediction models are boosting and bagging (for further information see: Witten and Frank (2000), Han et al. (2005)). As an example of prediction based on game metrics (specifically game-play metrics), we will use *Tomb Raider: Underworld* (Eidos Interactive, 2008) e. For a more in-depth description of the example, please see Mahlman et al. (2010).

Tomb Raider: Underworld consists of seven levels plus a prologue level. The goal of this analysis was to investigate if it was possible to develop a model that could predict when a player would stop playing the game, based on their early play behavior. This kind of prediction is useful to locate players who stop playing early in the game, and explore why this happens and how to modify the design to prevent players from leaving the game. For this experiment, the Weka (www.cs.waikato.ac.nz/ml/weka/) toolbox was used. Weka is a relatively easy-to-use toolset for data mining, and includes a wealth of prediction algorithms 76 just for classification of nominal attributes – and it is open-source.

The data for the analysis was drawn from the native metrics suite of Square Enix Europe, which contains data from approximately 1.5 million players of *Tomb Raider: Underworld*. From this population, a sub-sample of 10,000 players was

selected, randomly drawn from a larger sample of over 200,000 players, from which the metrics data was captured within the period of 1st December 2008–1st January 2009. The original sample of 10,000 players was cleaned thoroughly, removing instances where the metrics suite had missing data reported for a player. Because the aim of the analysis was to predict when players stop playing the game, only players who had completed level one were included. After cleaning the 10,000 player sample, 6,430 players remained. The data from level 1 were used as the training (learning) dataset. All features were normalized to a 0–1 scale via a uniform distribution to minimize the effect of outliers.

The input features (variables) were selected from the core mechanics of the game, a strategy which helps with ensuring that the features are relevant to player behavior analysis. Each feature was measured either per map unit or per level, giving a total feature set of over 400 variables (number of features * level/map unit):

- **Playing time:** the time that each player spent playing the game. This includes a number of features, notably the playing time spent for each sub-segment of each level in the game (there are over 70 such segments).
- **Total number of deaths:** how many times the player died.
- **Help-on-Demand:** how many times the player requested help from the native Help-on-Demand system in the game, which assists players with their progress in the game.
- **Causes of death:** the game features various ways in which a player can die. These were classified into four groups: Death by melee enemies, death by ranged enemies, death by environmental causes, and death by falling (by far the most common cause of death in Tomb Raider: Underworld – 62.92%).
- **Adrenalin:** the number of time the adrenalin feature was used. Using adrenalin allows the player to temporarily slow down time while performing special attacks.
- **Rewards:** the number of rewards collected (the average is 112.08). Treasure: The number of treasures found. Each level has one or a few of these major finds, which take particular exploration to locate.
- **Setting changes:** players can change various parameters of the game, and four of these impact directly on gameplay and were therefore included: Ammo adjustment, enemy hit points, player hit points, and saving grab adjustment (which adjusts the time a player has to secure a handhold after a jump).

From the dataset of 6,430 players (including all the variables mentioned above), who completed level 1 at the least, a smaller dataset was extracted which consisted of the 3,517 players who also completed at least level 2. A third set of data was created from the second, containing the 1,732 players that finished the entire game. The three datasets were used to try to predict the time taken to play through the game, with the underlying assumption that there is an (unknown) relationship or function between early playing behavior (levels 1 and 2) and the speed with which a game is completed, or conversely when a player will stop playing (i.e. last level played), which a classification algorithm will be able to predict.

Several classification algorithms were used on the two problems: completion time and last level played. The best results were found using logistic regression, a relatively simple algorithm which could predict when a player would stop playing *Tomb Raider: Underworld*, with a success rate of 77.3%. Several algorithms performed well on the dataset (notably SMO support vector machine, MLP/Backpropagation), with a much better accuracy than the baseline of 39.8% (the baseline is the optimal predictor in case there is no data available, equal to the number of samples in the most common class (level completed) divided by the total number of classes). The accuracy of the prediction is in this case decent. Typical prediction models were built on high-dimensionality gameplay metrics dataset (in this case hundreds of features), presumably due to either the high degree of variance in the datasets, i.e. in how people play games, and data losses during collection of telemetry data from game clients (Drachen et al. 2009).

The ability to predict when a player will stop playing a game, or for how long the game will be played, based on their early behavior is useful in user-oriented testing, where it is possible to use this information to locate the kinds of behaviors that lead players to quit playing. This is particularly useful in certain forms of social online games, where player retention (the ability of the game to keep people playing it) is central to the revenue stream (see Chap. 4).

12.4.2 Decision Trees

Results from prediction analysis need to be explained in a way that makes them understandable to the target user, e.g., a game designer. Apart from accuracy in the predictions, an advantage of some of the algorithms for predictive data mining is that they provide relatively transparent models, which means that changes to design elements can be easily understood.

Decision trees are a good example of this. They use a graphic approach to compare competing alternatives, and assign values to these alternatives, describing sequential decision problems. They provide a complementary approach to traditional statistical forms of analysis such as multiple linear regression and data mining approaches such as neural networks. They are relatively powerful analytically, easy to use, easy to interpret and robust within a range of data and levels of measurement. They are presented incrementally, in a collection of one-cause, one-effect relationships in the recursive form of a tree, which means they are easy to understand than more complex multiple variable techniques (Rokach and Maimon 2008).

Like other methods of multiple variable analyses, they allow the prediction, explanation, description or classification or an outcome. For example, a multiple variable analysis could be the probability that a player will convert from non-paying to paying as a result of the combined effect of multiple variables, e.g. a marketing campaign, being given a valuable in-game item, the size of their social network in

the game – or being given a free T-shirt if they sign up for a subscription. In essence, decision trees allow analysts to follow the effect of different decisions, and plan the optimal strategy for causing specific decisions (or situations) to occur in the games in question. For example, answering questions such as: which set of methods for encouraging player to become paying users work the best?

Decision trees are produced by algorithms, which try to split a dataset into branch-like segments – hence the name. The branches form inverted decision trees that originate with a root node at the top, and branch out from there. Decision trees attempt to find relationships between input values and target values in a dataset (group of observations). When an input value is identified as being strongly related to a target value, all of these variables are grouped in a bin that becomes a branch in the tree. A strong relationship is formed when the value of an input variable can be used to predict the value of the target. For example, if the amount of time a player spends in a particular map unit of *Tomb Raider: Underworld*, is a strong predictor of the completion time of the entire level. Or the number of times a player activates the adrenaline feature of the game (an advanced game mechanic) could be a predictor of whether the player is an experienced player or not.

To take a hypothetical example from *Tomb Raider: Underworld* (see Mahlman et al. 2010 for a more in-depth example), where decision tree analysis is employed to predict which level players will stop playing at, as a feature of playtime and rewards, the resulting tree could look like this:

```

Level-2 rewards
Rewards > 10
Level-3 playtime
    ↳playtime > 43 minutes : 4
    ↳playtime < 43 minutes : 7
Rewards < 10 : 2

```

The right arrow (→) indicates a branch under the tree-node, which is directly above the symbol. The number to the right of the colon represents the predicted game level where the player will stop playing. What the tree means is that a strong relationship has been found between the level at which players stop playing, and the rewards earned at level 2, and the playtime at level 3. The first branch informs that if the player earns less than 10 rewards, they will stop playing at level 2. The second branch informs that if the players spend more than 43 min on level 3, they will stop playing on level 4, but if they complete below this time, they will play through the entire game.

Decision trees like this one can be employed on virtually any kind of variable tracked via telemetry, i.e. behavioral variables, in order to find out which features are the most important to determine when a player quits playing (or any other outcome being measured, e.g. how much money they spend on microtransactions) (Chap. 4), and the values of these features to prompt different end-points in the tree.

12.5 Game Data Mining in Free-to-Play Games

In this section we take a specific look at online games. These games are of particular interest in game data mining because they are highly dependent on understanding player behavior, and currently one of the major forces driving the use of and innovation in data mining in game development. It is not in the goal here to provide a comprehensive introduction to data mining in online games, but a brief overview. The reader is referred to the references for additional information.

To start with a brief (and generalized) historical perspective, the current push for data mining player behavior in the industry has to a certain extent been driven by the rise of the social online game – or Free-to-Play (F2P) – genre, as well as the widespread popularity of the Massively Multiplayer Online Game (MMOG) genre. MMOGs have an almost two decades long history, reaching back to games like *Meridian 59* and *EverQuest*. They started getting serious attention in the regular press with the realization that these online, persistent worlds contained intricate economies (Castranova 2001), and with *Second Life* (arguably a virtual world, not a game) and notably *World of Warcraft*, that they had become highly popular. With the evolution of Web 2.0 technologies, notably social networking platforms like Facebook, another type of game also increased in popularity: F2P, with early examples on Facebook including *Mafia Wars*.

MMOGs and F2P games were different from previous game forms in that they catered to very large groups of players who could interact in real time. MMOGs and many F2Ps are also persistent world games – they are always running – which facilitated the emergence of social communities in these games.

In the past few years metrics-driven development has almost become standard in online games development and –management. Acronyms and terms like ARPU, NOSQL and Big Data are becoming commonplace (see Chap. 4), and it is likely that most publishers and developers in the online games sphere are highly dependent on analytics and reports to keep their businesses running. While many Key Performance Indicators (KPIs) are common (Fig. 12.12), the level of sophistication in the analytics software and processes vary across the industry (Flood 2012). Competition, the cross-over of players between different sectors of the games industry, and the evolution in player communities over time, requires online games companies to field efficient data capture and storage, and the ability to generate KPIs and ad-hoc analysis and reporting.

12.5.1 Metrics-Driven Business Practices in Online Games

A lot more could be said about the historic background for MMOGs and F2Ps (for more information see Fields and Cotton 2011), but the essence of the matter is that these games need data mining because they have to *manage and monetize on a community of players*. Generalizing, the essential requirement in the MMOG business model is to keep people engaged so they continue to pay subscription fees.

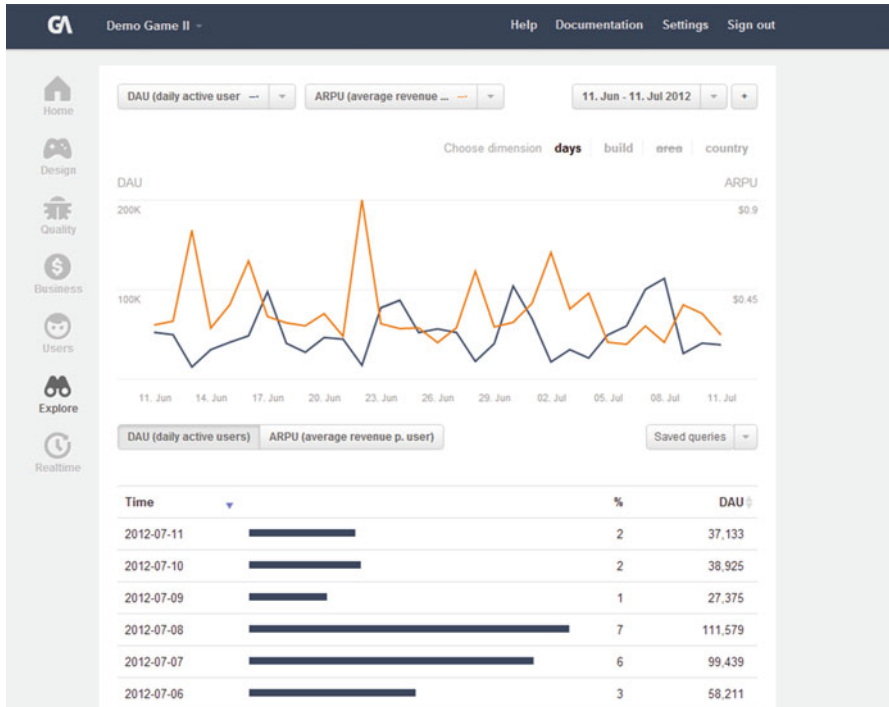


Fig. 12.12 Screenshot from the an early beta version of the analytics tool from Game Analytics, showing the development over time of two of the common Key Performance Indicators (KPIs) for online games: Daily Active Users (DAU) and Average Revenue Per User (ARPU) (© Game Analytics, used with permission, www.gameanalytics.com)

The requirement for F2P games is to convince players to spend money on buying in-game resources.

There are a number of ways to handle this kind of challenge, but fundamentally relate to Business Intelligence management. There are a number of similarities between managing and monetizing on player communities and the management of websites, online forums and web-based communities in general. These, similar to online games, have customers coming and going, interacting with the site and/or people via the site, for shorter or longer periods of time.

Web analytics is the field of research and practice dealing with quantitative analysis of user behavior on the Net (Jansen 2009), and back when MMOG and F2P models were gaining momentum, there was a lot of knowledge available that could be adapted for use in these – and other – types of games, for example with regards to online advertising and customer retention, and the use of techniques like funnel analysis and cohort analysis to understand the cost of acquisition, retention factors, revenue generation, social factors, etc.

The metrics-driven business practice in online games gained strong traction with the rapid growth of game companies like Zynga, BigFish and Wooga, who had

adopted a metrics-driven development practice and became highly successful in a short period of time, and the growth of the social application market in general (e.g. Facebook, InstaGram, LinkedIn, Google+, Twitter, Picasa ...). Business intelligence has emerged as a key aspect of operating a successful online games company.

12.5.2 *Data Mining Telemetry from Online Games*

Whereas the publicly available knowledge about data mining in MMOGs is limited due to confidentiality issues, the available knowledge for F2P games is more comprehensive if somewhat fragmented, but generally originates in articles, blog posts or reports from the industry, and is therefore not falsifiable. With that in mind, the data mining techniques for analyzing player telemetry from online games can be broadly divided into three broad categories:

1. **Key Performance Indicators:** These are metrics like Daily Active Users (DAU) and Churn rate, which are generated using descriptive methods, e.g. aggregates or ratios, typically calculated as a function of time, game build or geographical area (Chap. 4; Fields and Cotton 2011).
2. **Adopted techniques:** These are techniques adopted – and sometimes subsequently adapted – from other areas where Business intelligence is applied, notably web analytics. Examples include acquisition analysis, funnel analysis, A/B testing and cohort analysis (Chap. 4; Fields and Cotton 2011). These methods are generally descriptive or examples of characterization and discrimination.
3. **Advanced techniques:** These are techniques that rely on data mining methods for clustering, classification, prediction, estimation and association. Notably user behavior prediction (Weber et al. 2011; Nozhnin 2012; Bauckhage et al. 2012; Lim 2012), classification of user behavior (Drachen et al. 2009, 2012) and retention modeling (Fields and Cotton 2011) has received interest in the online games sector, as these techniques are of key interest in driving revenue.

12.6 Discussion and Next Steps

In this chapter, we have presented an introduction to data mining and its particular application in game development, *game data mining*. A number of important issues in relation to working with game telemetry datasets have been discussed, covering topics such as methods, stakeholders and practice. Additionally, we have outlined a number of examples showing how to perform different types of supervised and unsupervised analysis on game telemetry data.

While the focus of the chapter is on game telemetry data, and the types of problems they can be applied to solve, the general principles and the methods presented are not unique, but rather common in data mining across several application areas, and therefore accessible in a wealth of literature to anyone interested in learning more about data mining (e.g. Han et al. 2005). For more on game data mining,

Chap. 4 outlines KPIs for online games; Chap. 7 describes developer-facing analytics, Chap. 17 discusses game data mining in the specific context of spatial data, i.e. data with a spatial component (e.g. data on player movement in a 3D environment). Chapter 18 and 19 go into more depth with visualization of game telemetry data.

Game telemetry presents some challenges that are uncommon or maybe even unique in large-scale user-oriented datasets:

1. The data can have a high dimensionality, often with thousands of features (or variables) being tracked for each user.
2. The data can be of substantial size, an average MMOG or social online game generating datasets on the terabyte scale.
3. It is often necessary to compile datasets for analysis from disparate sources, e.g., game telemetry and account systems, with associated challenges in merging data and avoiding redundancies.
4. Obtaining game telemetry from remote clients, across multiple hardware platforms (many games are released on multiple hardware platforms), requires well-designed back-end systems to ensure that the datasets are as complete as possible. This is in particular a challenge when collecting data from devices that are not online all the time while the user is playing, e.g. games for smartphones.

The list goes on, and it is out of scope here to provide a full discussion of all of the issues related to game data mining. However, we provide a starting point for non-experts, and hopefully some case studies that will also satisfy the game data mining expert as well.

It can seem like a daunting project to develop both the technical back-end for collecting and storing game telemetry, as well as learning how to pre-process and subsequently analyze the datasets, and finally finding the best ways to present the results to various stakeholders in development companies. The cost alone can seem prohibitive, but the simple fact that publishers like Microsoft, Square Enix, EA Games and Ubisoft are employing game telemetry, the success stories of companies like Zynga, Wooga, Bigfish and Bungie, are strong indications of the benefits that can be obtained via game data mining.

To make game data mining easier, there is these years a proliferation of tech startups seeking to develop middleware tools that enable even small developers to work with telemetry data (e.g. Game Analytics, Honeytracks, Playtomic, Playnomics, Tableau, Kontagent), as well as a number of open-source tools for different engines (e.g. Unity) and analytics packages that can be used for analyzing game telemetry (e.g. statistics packages like SPSS, open-source data mining tools like WEKA and RapidMiner). While the main focus has hitherto been on analyses of player behavior (e.g. Zoeller 2010) (see also Chaps. 4, 7, 14, 17, 18, and 19) and customer data (e.g. King and Chen 2009) (see Chap. 4), the potential scope of application of game data mining as a source of business intelligence is substantial, crossing marketing, production (Mellon 2009) (see Chaps. 6 and 7), design, user testing, strategic decision making, etc., and if the current rapid development in the application of various types of measures to guide game development including game telemetry is any indication, the application of game metrics in the digital

entertainment industry will become a standard that is as normal as other types of processes in business, e.g. benchmarking and usability testing – if you need more evidence, read any other chapter in this book.

About the Authors

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Christian Thureau, Ph.D. is CTO of Game Analytics (www.gameanalytics.com) and a former researcher at the Fraunhofer IAIS in St. Augustin and at the Bonn-Aachen International Center for Information Technology B-IT. He works with developing game telemetry systems and the application of advanced data mining methods in games contexts, for example player modeling, behavior cloning and data analysis in the massive dataset size range. His fields of research include Pattern Recognition, Computer Vision, Data Mining, and Machine Learning.

Julian Togelius, Ph.D. is Associate Professor at the Center for Computer Games Research, IT University of Copenhagen, Denmark. He works on all aspects of computational intelligence and games, on geometric generalization of stochastic search algorithms and on evolutionary reinforcement learning. His current main research directions involve search-based procedural content generation in games, automatic game design, and fair and relevant benchmarking of game AI through competitions. He is also the chair of the IEEE CIS Technical Committee on Games, and an associate editor of IEEE Transactions on Computational Intelligence and Games. He holds a BA from Lund University, an MSc from the University of Sussex, and a PhD from the University of Essex.

Georgios N. Yannakakis, Ph.D. is an Associate Professor at the IT University of Copenhagen. He received the Ph.D. degree in Informatics from the University of Edinburgh in 2005. Prior to joining the Center for Computer Games Research, ITU, in 2007, he was a postdoctoral researcher at the Mærsk Mc-Kinney Møller Institute, University of Southern Denmark. His research interests include user modeling, neuro-evolution, computational intelligence in computer games, machine

learning, cognitive modeling and affective computing. He has published over 100 journal and international conference papers in the aforementioned fields. He is an Associate Editor of the IEEE Transactions on Affective Computing and the IEEE Transactions on Computational Intelligence and AI in Games, and the chair of the IEEE CIS Task Force on Player Satisfaction Modeling.

Christian Bauckhage, Ph.D. is professor of media informatics at the University of Bonn and lead scientists for multimedia pattern recognition Fraunhofer IAIS. He obtained a PhD in computer Science from Bielefeld University, Germany, and worked in academic and industrial research labs. He is an expert in large scale data mining and pattern recognition and researches computational approaches to artificial cognition and behavior.

References

- Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM-SIGMOD international conference on management of data (SIGMOD)* (pp. 207–216). Washington, DC.
- Bauckhage, C., Kerstin, C., Sifa, R., Thurau, C., Drachen, A., & Canossa, A. (2012). How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *Proceedings of IEEE computational intelligence in games*, Granada, Spain.
- Berry, M., & Linoff, G. (1999). *Mastering data mining: The art and science of customer relationship management*. New York: Wiley.
- Bishop, C. M. (2006). *Pattern recognition and machine learning* (Information science and statistics). New York: Springer.
- Bohannon, J. (2010). Game-miners grapple with massive data. *Science*, 330(6000), 30–31.
- Castranova, E. (2001). *Virtual worlds: A first-hand account of market and society on the Cyberian frontier* (CESifo Working Paper Series no 618). München.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinart, T., Shearer, C., & Wirth, R. (2000). Crispdm step-by-step data mining guide. <http://www.crisp-dm.org/>
- Charles, D., & Black, M. (2004, November 8–10). Dynamic player modelling: A framework for playercentric digital games. In *Proceedings of CGAIDE 2004, 5th international conference on computer games: Artificial intelligence, design and education*. Microsoft Campus, Reading, UK. ISBN 09549016-0-6
- Chen, M. S., Han, J., & Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8, 866–883.
- Coulton, P., Bamford, W., Cheverst, K., & Rashid, O. (2008). 3D space-time visualization of player behavior in pervasive location-based games. *International Journal of Computer Games Technology Volume 2008 (2008)*, Article ID 192153, 5 pages. doi:10.1155/2008/192153
- Cutler, A., & Breiman, L. (1994). Archetypal analysis. *Technometrics*, 36(4), 338–347.
- DeRosa, P. (2007, August 7). Tracking player feedback to improve game design. Gamasutra. Available from: http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_php
- Drachen, A., & Canossa, A. (2009). Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th international MindTrek conference*. Tampere: ACM.
- Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behavior in virtual environments. *International Journal of Arts and Technology*, 4, 294–314.
- Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the international symposium on Computational Intelligence and Games, CIG'09*, Piscataway.

- Drachen, A., Sifa, R., Bauckhage, C., & Thurau, C. (2012). Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Proceedings of IEEE computational intelligence in games*, Granada, Spain.
- Ducheneaut, N., & Moore, R. J. (2004). The social side of gaming: A study of interaction patterns in a massively multiplayer online game. In *Proceedings of the 2004 ACM conference on computer supported cooperative work*, Chicago.
- Erfani Joorabchi, M., Seif El-Nasr, M. (2011, October, 5–8). Measuring the impact of knowledge gained from playing FPS and RPG games on gameplay performance. In *Proceedings of 10th international conference, ICEC 2011* (Lecture notes in computer science, Vol. 6972, pp. 300–306). Vancouver.
- Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Menlo Park: AAAI Press.
- Fields, T., & Cotton, B. (2011). *Social game design: Monetization methods and mechanics*. Waltham: Morgan Kaufman Publishers.
- Finesso, L., & Spreij, P. (2004). Approximate nonnegative matrix factorization via alternating minimization. In *Proceedings 16th international symposium on mathematical theory of networks and systems*, Leuven.
- Flood, K. (2012, March 27). Game analytics (series). Kevin's corner. URL: file:///G:/Work/METRICS/Metrics_references/Kevin%27s%20Corner%20%20Game%20Analytics.htm
- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2012). Analysis of telemetry data from a real time strategy game: A case study. *Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment*, 10(3), Article No. 2. New York: ACM. doi:10.1145/2381876.2381878.
- Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), Article No. 9. New York: ACM. doi:10.1145/1132960.1132963.
- Golub, G., & van Loan, J. (1996). *Matrix computations* (3rd ed.). Baltimore: Johns Hopkins University Press.
- Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM-SIGMOD international conference on management of data (SIGMOD)* (pp. 1–12). New York.
- Han, J., Kamber, M., & Pei, J. (2005). *Data mining: Concepts and techniques* (Morgan Kaufmann large-scale data mining in games 41 2nd ed.). San Francisco: Morgan Kaufmann Publishers.
- Hoobler, N., Humphreys, G., & Agrawala, M. (2004). Visualizing competitive behaviors in multi-user virtual environments. In *Proceedings of the conference on visualization*. Los Alamitos: IEEE.
- Houlette, R. (2004). Player modeling for adaptive games. In S. Rabin (Ed.), *AI game programming wisdom II* (pp. 557–566). Hingham: Charles River Media.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco: Morgan Kaufman.
- Jansen, B. J. (2009). *Understanding user-web interactions via web analytics*. San Rafael: Morgan & Claypool Publishers.
- Jolliffe, I. (1986). *Principal component analysis*. New York: Springer.
- Kastbjerg, E. (2011). *Combining sequence mining and heatmaps to visualize game event flows (working title)*. Master's thesis, IT University of Copenhagen, Copenhagen.
- Kennerly, D. (2003, August 15). Better game design through data mining. Gamasutra. Available from: http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_.php
- Kim, J. H., Gunn, D. V., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of the twenty-sixth annual SIGCHI conference on human factors in computing systems, CHI'08*, Florence.
- King, D., & Chen, S. (2009). Metrics for social games. *Presentation at the social games summit 2009, game developers conference*. San Francisco, CA.
- Larose, D. T. (2004). *Discovering knowledge in data: An introduction to data mining*. Hoboken: Wiley.
- Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 788–799.

- Lim, N. (2012, June 26). Freemium games are not normal. Gamasutra. URL: http://www.gamasutra.com/blogs/NickLim/20120626/173051/Freemium_games_are_not_normal.php?goback=.gmr_4199042.gde_4199042_member_130240768.gmr_4199042.gde_4199042_member_128990050#comments
- Mahlman, T., Drachen, A., Canossa, A., Togelius, J., & Yannakakis, G. (2010). Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the international conference on Computational Intelligence and Games, CIG'10*, Copenhagen.
- Mellon, L. (2009). *Applying metrics driven development to MMO costs and risks*. White paper, Versant Corporation.
- Missura, O., & Gärtner, T. (2009). Player modeling for intelligent difficulty adjustment. In *Proceedings of the 12th international conference on discovery science, DC'09*, Berlin.
- Moura, D., Seif El-Nasr, M., & Shaw, C. D. (2011). Visualizing and understanding players' behavior in video games: Discovering patterns and supporting aggregation and comparison. In *Proceedings of the 2011 ACM SIGGRAPH symposium on video games (Sandbox '11)* (pp. 11–15). New York. ISBN:978-1-4503-0775-8, doi:10.1145/2018556.2018559.
- Nozhnin, D. (2012, May 17). Predicting churn: Data-mining your game. Gamasutra. URL: http://www.gamasutra.com/view/feature/170472/predicting_churn_datamining_your_php
- Paatero, P., & Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2), 111–126.
- Pedersen, C., Togelius, J., & Yannakakis, G. N. (2010). Modeling player experience for content creation. *Transactions on Computational Intelligence and AI in Games*, 2, 54–67.
- Rokach, L., & Maimon, O. (2008). *Data mining with decision trees: Theory and applications*. New Jersey: World Scientific Publishing.
- Seif El-Nasr, M., & Zammito, V. (2010). User experience research for sports games. *Presentation at the GDC summit on games user research*. San Francisco, CA.
- Seif El-Nasr, M., Aghabeigi, B., Milam, D., Erfani, M., Lameman, B., Maygoli, H., & Mah, S. (2010). Understanding and evaluating cooperative games. *CHI 2010* (pp. 253–262). New York.
- Shaker, N., Yannakakis, G., & Togelius, J. (2011). Feature analysis for modeling game content quality. In *Proceedings of the 2011 IEEE conference on computational intelligence and games* (pp. 126–133). Seoul, Korea
- Summit Kohonen, T. (2001). *Self-organizing maps*. Heidelberg: Springer.
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Adaptive computation and machine learning). Cambridge: The MIT Press.
- Thawonmas, R., & Iizuka, K. (2008). Visualization of online-game players based on their action behaviors. *International Journal of Computer Games Technology*, 2008, 1–9.
- Thawonmas, R., Kashifuji, Y., & Chen, K. T. (2008, December 3–5). Design of MMORPG Bots based on behavior analysis. In *Proceedings of the 2008 international conference on advances in computer entertainment technology, ACE'08*, Yokohama, Japan (ACM International Conference Proceeding Series 352, pp. 91–94). doi:10.1145/1501750.1501770, ISBN:978-1-60558-393-8.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine*.
- Thureau, C., & Bauckhage, C. (2010). Analyzing the evolution of social groups in world of warcraft. In *Proceedings of the international conference on Computational Intelligence and Games, IEEE, CIG'10*, Copenhagen.
- Thureau, C., & Drachen, A. (2011). Introducing archetypal analysis for player classification in games. In *Proceedings of the international workshop on evaluating player experience in games (EPEX'11) hosted at the 6th international conference on the foundations of digital games (FDG2011)*. Bordeaux.
- Thureau, C., Bauckhage, C., & Sagerer, G. (2004, July 13–17). Learning human-like movement behavior for computer games. In *Proceedings of the 8th international conference on the Simulation of Adaptive Behavior, SAB'04*. Los Angeles, USA. ISBN: 9780262693417.
- Thureau, C., Paczian, T., Sagerer, G., & Bauckhage, C. (2007). Bayesian imitation learning in game characters. *International Journal of Intelligent Systems Technologies and Applications*, 2(2–3), 284–295.

- Thurau, C., Kersting, K., & Bauckhage, C. (2009). Convex non-negative matrix factorization in the wild. In *Proceedings of the IEEE international conference on data mining*, Miami.
- Thurau, C., Kersting, K., & Bauckhage, C. (2010). Yes we can – Simplex volume maximization for descriptive web-scale matrix factorization. In *Proceedings of the international Conference on Information and Knowledge Management, ACM, CIKM'10*, Toronto.
- Thurau, C., Kersting, K., Wahabzada, M., & Bauckhage, C. (2011). Descriptive matrix factorization for sustainability: Adopting the principle of opposites. *Journal of Data Mining and Knowledge Discovery*, 24, 325–354.
- Weber, B., & Mateas, M. (2009). A data mining approach to strategy prediction. In *Proceedings of the international symposium on Computational Intelligence and Games, CIG'09*, Piscataway.
- Weber, B. G. John, M. Mateas, M. & Jhala, A. (2011). Modeling player retention in Madden NFL 11. In *Proceedings of the association for the advancement of artificial intelligence conference*, San Francisco.
- Williams, D., Yee, N., & Caplan, S. E. (2008). Who plays, how much, and why? Debunking the stereotypical gamer profile. *Journal of Computer-Mediated Communication*, 13, 993–1018.
- Williams, D., Consalvo, M., Caplan, S., & Yee, N. (2009). Looking for gender (LFG): Gender roles and behaviors among online gamers. *Journal of Communication*, 59, 700–725.
- Witten, I. H., & Frank, E. (2000). *Data mining*. New York: Morgan-Kaufmann.
- Yannakakis, G. A. (2012). Game AI revisited. In *Proceedings of the conference on computing frontiers*, Caligari.
- Yannakakis, G. N., & Hallam, J. (2009). Real-time game adaptation for optimizing player satisfaction. *Transactions on Computational Intelligence and AI in Games*, 1, 121–133.
- Yannakakis, G. A., & Togelius, J. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, 2 (3), 147–161
- Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42, 31–60.
- Zoeller, G. (2010). Game development telemetry. *Presentation at the game developers conference 2010*.

Chapter 13

Meaning in Gameplay: Filtering Variables, Defining Metrics, Extracting Features and Creating Models for Gameplay Analysis

Alessandro Canossa

Take Away Points:

1. *Defining a vocabulary for mapping game systems into measurable variables.*
2. *Unfolding the process of creating features to be used in modeling player behavior.*
3. *Drafting two strategies to create models of player behavior: top-down models utilizing designer-driven play personas and bottom-up models utilizing algorithm-driven computational models.*

13.1 Introduction

Analyzing game-related data, at its core, is a process that involves being able to articulate knowledge and meaning from apparently meaningless data. Analysis often consists of imposing order, establishing categories and seeing patterns in disorderly, continuous and heterogeneous streams of information, especially when dealing with gameplay telemetry data, which directly emanates from players' behavior. Since human behavior represents the response of an organism to its ecosystem, it possesses no intrinsic meaning; rather it needs to be interpreted. It is mostly through interpretation of data that actionable knowledge and pertinent meaning can be massaged into existence according to the assumption that player motivations, desires, beliefs and personality are encoded in a player's behavior

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

and it is sufficient to interpret metrics data to unravel extensive information about players. Ludwig Wittgenstein, in his *Tractatus logico-philosophicus*, (Wittgenstein 2001) said that “*the limits of my language mean the limits of my world*” implying that the logical possibilities available within a certain domain are constrained by the language used to talk about such a domain. In the specific case of game data analysis, the verbs used to talk about player behavior are defined by the game variables measured and tracked by the telemetry system. These variables, once measured, become metrics, and from metrics, features are extracted; the selection of which features to use is a pivotal component of game data analysis. This chapter presents strategies to aid in this process, specifically, in the selection of variables, their measurement and the treatment of the resulting features to obtain meaningful models. The process of selecting game variables to be monitored for further analysis is not a trivial one since it is exactly this process of selection that defines which analyses can be carried out and enables analysts to draw inferences from the game.

13.1.1 Filtering What to Track

The holistic approach, which is to monitor every possible game variable, has several drawbacks. Even if it is tempting, “choosing not to choose” is a very risky practice for a number of reasons.

Every variable that needs to be tracked requires time to be coded, a process often in the hands of programmers and programmer time is a precious commodity during development; hence a careful selection could limit the amount of resources that programmers have to devote to telemetry system setup.

Furthermore there are also logistic limitations in the form of bandwidth and storage constraints: every bit of information gathered from players needs to be transferred and stored. For example, during the multiplayer beta test for *Halo Reach* (Microsoft, 2010), around 2.7 million players participated over a period of 17 days, generating 16 million total hours of play time and several terabytes of data. The developers chose to track many dozens of variables. According to Marcus Letho, Bungie’s Creative Director, having a large audience of players “hammering” on the game, allowed the developers to gather useful feedback to fix bugs and fine tune gameplay (Sydney Morning Herald 2010). Brian Jarrard, Bungie’s community manager, also claimed that the large beta was vital to seeing how the game would perform (Brudvig 2010). On the other hand, understandably, handling this heavy traffic was a serious challenge. For company less prepared and with fewer resources than Bungie, the data collection session could have been a total failure.

Another limit imposed on the number of features tracked is determined by data retrieval and query execution time. More often than not, the collected game telemetry data resides on remote servers and needs to be transferred to the local client performing the analysis. A dataset containing just 12 features for two million

Tomb Raider: Underworld (Eidos Interactive, 2008) players over the course of 3 months totaled slightly over 4 terabytes. Large datasets such as this are not just difficult to handle during collection from players' client, as seen in Bungie's *Halo Reach*, but can be extremely time consuming also during retrieval and query execution time, in fact simple queries with few columns in the SELECT statement and few filters in the WHERE clause, caused waiting times in excess of 1 hour.

The resources available at the development studio also represent a very tangible constraint on features collected: not so much in terms of what gets collected, but more in terms of how much. If there is nobody employed to carry out complex analyses on the dataset, maybe it's pointless to collect highly granular data about all sorts of game variables, conversely, studios that invested considerably in data mining personnel might afford to err on the side of abundance, collecting more than really necessary.

Eventually, there is also a limit on the amount of information the human brain can successfully process: "information overload" refers to the excess of information up to the point where it becomes impossible to use this information effectively. Information stops forming a scaffold to decision-making and it becomes an obstacle instead (Edmunds and Morris 2000).

Not only is necessary to filter the number of variables, but it is imperative to possess complete clarity on the level of detail and the granularity of the variables that are to be tracked before starting to gather data and before playtests are run; the price to pay for deciding to include a new variable after a battery of playtests is the necessity to repeat all the tests. Utilizing the tool of game metrics, as many other aspects of game development, is an iterative process where each cycle has a cost in terms of time and resources that needs to be scheduled for.

Considering the hurdles involved in collecting the right data, several developers started looking at dynamic metrics tracking as a possible solution. Dynamic metrics tracking refers to the possibility of deciding which variable to measure at any time among all the possible variables. This requires a system able to seamlessly change which metrics are collected from the players' clients by remotely operating on the game. If that was possible, developers would no longer be required to make a decision a priori and commit to that for good, but would be able to decide impulsively what to monitor without overloading bandwidth and storage to critical levels. This non-selective approach is in fact technically feasible and it is already used extensively, often in cooperation with A-B testing practices, by developers focusing on casual markets for personal computers and mobile computing devices (Pincus and Gordon 2009). Unfortunately, dynamic metrics tracking will have to remain an elusive chimera on consoles, at least for the time being. When a game is submitted for approval to Microsoft, Sony and Nintendo, the console manufacturers require a list of possible information exchanged between players and developers for legal and privacy issues. This means that any time a developers would like to change remotely the pool of features collected, the console manufacturer would need to re-approve the change in a manner similar to the submission evaluation of a game patch or downloadable content.

13.1.2 *Hypotheses Testing*

As it will be shown in Sect. 14.3, if there are concrete questions already set forth, the process of selecting game variables, measuring them and extracting features to answer the questions imposed by the stakeholders is fairly straightforward. The question itself often contains the necessary information to isolate the relevant variables. Obviously, it is still not a deterministic relation since deciding how to measure a phenomenon has intrinsic implications for the type of answers provided.

For example, if the question is “what is the most common cause of death” immediately analysts know which game systems need to be monitored, i.e. hostile NPCs, environmental hazards and self-inflicted damage. However, when deciding how to measure and log the cause of death, there are several alternative strategies:

- labeling every single possible cause that can lead to player death. Every class of hostile NPCs, as defined in the code, will constitute a different cause: yeti, sniper, spider, etc.
- grouping similar causes of death according to parameters dictated by domain knowledge and creating classes for possible causes of death: ranged-attack NPCs (of which sniper is one among many), melee NPCs (of which yeti is one among many), trap-setting NPCs (of which spider is one among many), etc.
- creating even larger and more comprehensive classes, not based on the meaning of the class for a human observer, but instead based on the structure and function of the underlying system. In this scenario, for example, trap-setting NPCs could be grouped with environmental hazards since, functionally, a static poisonous web set by spiders shares more features with a pool of lava than with a yeti that seeks and chases his target.

Hence, as it can be seen from these examples, there usually is a continuum of possibilities: from the raw, unprocessed data, captured as changes occurred to the classes of object as defined in the code, to more aggregated classes defined by expert domain knowledge.

If in doubt, the best strategy is to log data in the format that is closest to the classes defined in the code without any aggregation. For example logging the player’s position every 10th of a second instead of a summary of time spent per sub-location. This strategy insures the least possible bias operated on the data; unfortunately, this strategy also risks levitating the amount of data logged, streamed and stored, incurring the problems mentioned in Sect. 13.1.1. For example, if the question is “where do players spend most of their time”, it is unrealistic to continuously log the coordinates of players’ location sampled every tenth of a second for millions of players. It is a much more viable solution to aggregate that data client-side and only send the information in terms of “time spent per sub location.” Evidently, this type of aggregation distances itself from raw data and includes large amounts of bias in the form of expert knowledge.

In fact, it will be level designers defining the concept of sub-locations according to their own hermeneutic lens, and those concepts might not be universally accepted as truths.

Summarizing: if an initial hypothesis exists, the individuation of relevant game variables is a fairly straightforward process even though the measurement of such variables is still biased.

13.1.3 Exploratory Analyses

On the other hand, if there is no initial hypothesis, even the initial selection of relevant variables becomes a critical step. This chapter will examine in detail the process of dissecting game systems to individuate relevant variables, defining strategies to measure those variables, extracting features from the measures and selecting important features to create models.

The exploratory approach aims at summarizing the main characteristics of a given dataset by inspecting it and attempting to formulate models that best describe the data. These models, in case of gameplay metrics, often take the form of clusters or sets of objects that share similar features. In other words, for example, in an exploratory data analysis of players' behaviors in the game *Tomb Raider: Underworld* (Eidos Interactive, 2008) we discovered four groups of players that face challenges in the game in a similar manner: veterans, solvers, pacifists and runners (Drachen et al. 2009).

Clusters of this kind are referred to as player types or player models. Selecting which features are relevant to create solid clusters is far from a trivial task; cluster analysis is an iterative process, as discussed in Sect. 12.3.1. Furthermore the different algorithms used to create clusters have different understanding of what constitutes a group, see Sect. 12.3.2. The presence of features that are not relevant for the definition of player types affects the similarity measure, degrading the quality of the clusters found by the algorithm.

The purpose of the exploratory approach is to understand a certain subject matter by creating models for it. Models are constructed by simplifying or abstracting the more complex underlying reality upon which they are based. Models can take the form of probability distribution functions with predictive capabilities, such as “all players who avoid confrontations will quit the game by level 10”, or classes of descriptive interpretations under which a particular statement is true, such as “*pacifists* are all players that avoid antagonistic confrontations.”

This chapter will examine two alternative methods to identify variables, select features and derive models from game telemetry data. Section 13.3 showcases a designer-driven, top-down, manual process, while Sect. 13.4 attempts to map the possibilities available when adopting an algorithmic, bottom-up, automatic approach; this chapter barely scratches the surface of the algorithmic approach, more on this subject is found in Chap. 12.

13.2 Variables, Metrics, Telemetry, Features and Models

13.2.1 Variables

Games have often been described as collection of interlocked systems (Salen and Zimmerman 2003). A game variable consists of potentially any change in the game systems resulted from a player interaction within the game. These include: navigating menus, starting a multiplayer session, choosing a character, moving the avatar, collecting power ups, and navigating dialogue trees.

It is evident how all the game systems can generate a very large number of variables. For example, the initial game menu for the game *Gears of War 2* (Microsoft, 2008) (GoW2) contains these elements: “solo campaign”, “co-op campaign”, “training grounds”, “multiplayer”, “horde”, “deleted scene”, “war journal”, “credits”, “what’s up”, “options” and “downloadable content” which amounts to 1 variable with 11 possible values, and we haven’t even started the game yet. In this case the variable defines the operations that players can perform in the initial menu screen.

Recapitulating: variables are deterministically derived from the possible changes in the game state that result from the interaction of players with the game.

13.2.2 Metrics and Telemetry

Metrics are generated every time a variable is measured automatically, while game telemetry refers specifically to metrics data that has been automatically transmitted remotely from a game client to a server.

The act of measuring, in itself, operates a bias, potentially affecting the analysis and the interpretation of the data. For example, regarding players interacting with the initial menu in GoW2, analysts may be interested in tracking all the options that were highlighted, and the amount of time each option was highlighted for, before a selection was made. Therefore, the variable must be treated both as a sampled metric and a triggered one, as discussed in Sect. 13.1. The highlighted state is sampled every 10th of a second, generating a stream of data besides the discreet event generated every time a selection is performed.

Establishing streamlined pertinent measures for the variables within the game potentially decreases clutter and noise, simplifying considerably further analyses. At the same time, streamlining excessively the measures might preclude subsequent analyses. For example, regarding the variable “initial menu selection” in GoW2, the decision to compound the attributes related to *PvP interaction* (“co-op campaign”, “what’s up”, “horde” and “multiplayer”) and the attributes related to *marketplace operations* (“downloadable content” and “deleted scene”) reduces the attributes from 11 to 6. This compounding of attributes facilitates initial analyses but might render impossible later analyses aimed at detecting competitive or cooperative trends in player behavior (see Sect. 13.2.3 in this chapter).

Defining how to measure variables may produce features if the data structure is conceptually distant from the raw code that the variables are based upon. In order to understand this distance it is sufficient to think of the difference between logging player position as a stream sampled every 10th of a second (metric close to the game state) and a summarized log of the time spent per sub-location (feature constructed with designers' knowledge of space). This is emblematic of the fact that there is not always a clear cut division between metrics and features. To a certain degree, there is a continuum between metrics that are pure measures of raw changes in the game state, and features that already embody strong biases and interpretations. For example, if analysts want to monitor the location of players, it is simple enough to sample the x, y coordinates of the avatar every 10th of a second, so the variable "location" becomes a metric in the form of a stream of x, y coordinates lasting as long as the play session lasts. It does not make sense due to storage, bandwidth and processing constraints to stream location telemetry every few milliseconds (see Sect. 13.1.1). If analysts want to monitor the location of millions of players, this telemetry stream can easily generate terabytes of data in a very short time. Thus, it is necessary to devise strategies to compact the potentially enormous steam of data into a more manageable format. For example, analysts may decide to summarize the information on the client-side, and create telemetry data in the form of "time spent in sub-location." In this way they can still receive the needed detailed information about player location, but as a succession of triggered events rather than streams of time-stamped data. Such suggested triggered events can look like this:

```
Location: catacombs01 EntryTime: 13,40,32
Location: catacombs02 EntryTime: 15,23,17
Location: catacombs01 EntryTime: 21,50,04
Location: sewers01 EntryTime: 22,06,48
```

This strategy reduces a hypothetical log of 10 min of gameplay from 3,600 entries (sampled every 10th of a second) to barely 4. The obvious drawback is the loss in the granularity of the data, rendering creating precise heatmaps impossible. A fact that may escape an initial superficial evaluation of this information-compacting process is the bias operated by level designers when they are asked to define sub-locations. Designers will reproduce an understanding of the subdivision of space that can be seen as arbitrary, imposing order and discreteness in an undivided spatial continuum. Alternatively, the spatial continuum can be subdivided utilizing flow parameters, such as choke points, hubs, edges and paths, and the resulting subdivisions may not necessarily be identical to the sub-locations defined by level designers. As a matter of fact, designer-imposed spatial categories are often not arbitrary. In fact, they incorporate precious domain knowledge. For example, if we take the previous hypothetical compressed game log, the Catacomb area is composed of two parts conceptually different, the initial part is filled with environmental challenges, and the second part contains creatures and NPCs thus adding to the challenges that the player must face. From the technical standpoint, the catacombs are one single entity with no loading in between the two parts. However, from the navigation viewpoint, paths, choke points and edges do not split the catacombs in

Table 13.1 Attributes: “solo campaign”(1), “co-op campaign”(2), “training grounds”(3), “multiplayer”(4), “horde”(5), “deleted scene”(6), “war journal”(7), “credits”(8), “what’s up”(9), “options”(10) and “downloadable content”(11)

Date/Time	Attribute
02.01.2012/17.34.12	3
02.01.2012/17.41.18	10
02.01.2012/17.44.24	1
03.01.2012/10.34.51	1
04.01.2012/16.05.03	1
05.01.2012/21.15.48	4
05.01.2012/22.40.17	2
06.01.2012/17.33.21	1
06.01.2012/21.17.52	4

the same way that gameplay designers chose to distribute the different challenges for players. The bias introduced by designers is beneficial domain knowledge that discriminates space according to the challenges that populate it, rather than less subjective spatial criteria related to navigation flow. As mentioned earlier, this example shows how a simple variable, when measured and transmitted, does not necessarily only produce game telemetry, but it can provide features, defined as data structures that are conceptually distant from the raw code that the variables are based upon.

13.2.3 Features

Once variables are measured and turned into metrics data, it might be necessary to extract features. Features are the concrete values entered in complex datamining operations and used as input for different algorithms. They can represent heuristic perspectives on the metrics gathered; they form points of view, or interpretations, on the raw data. For example, in the case of the variable “initial menu selection” for the game Gears of War 2, if the only event measured is the selection performed, the relative metrics for an individual player would look like Table 13.1.

At this point it is possible to extract features, like “absolute favorite selection” which in this case would be “solo campaign” (option 1), or a feature like “most common sequence of selections” which in this case would be first “solo campaign” and then “multiplayer” (option 1 and 4), or again a feature like “least favorite selections” which in this dataset would return “horde”, “deleted scenes”, “war journal”, “credits”, “what’s up” and “downloadable content”.

Once several features are extracted, it is possible to start selecting those features, and only those features, that are sufficient to define groups of players with similar behaviors. For example, players with “absolute favorite selection” equal to 1 and “most common sequence of selections” equal to 1 and 4 can be clustered together in the “competitive” group, while players with “absolute favorite selection” equal to 2 and “most common sequence of selections” equal to 9 and 2 can be clustered

Table 13.2 Three features extracted from the 11 attributes of the single variable “initial menu selection”

Solo play	Social play	Extra info
Solo Campaign (1)	Co-op Campaign (2)	Deleted scenes (6)
Training Grounds (3)	Multiplayer (4)	War Journal (7)
	Horde (5)	Credits (8)
	What’s Up (9)	Options (11)

together in the “cooperative” group. The feature “least favorite selection” did not make a difference when creating these clusters. The “competitive” and the “cooperative” clusters are both descriptive and predictive models of player behavior.

13.2.4 Models

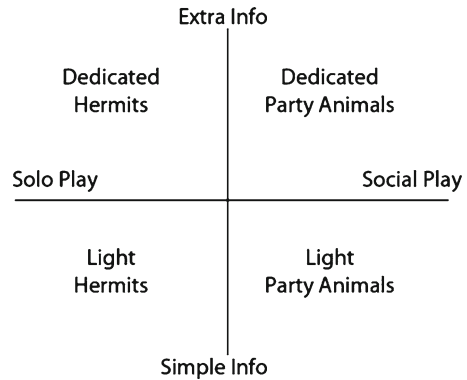
As hinted in Sect. 13.1.2, models are the culmination of exploratory analyses; they afford better understanding of the dataset at hand by either simplifying or abstracting the more complex underlying reality upon which they are based. Models can be extracted by classifying data and describing important classes or by predicting discrete and unordered labels. Models can be both descriptive interpretations under which a particular statement is true and also can be predictions of future data trends in continuous-valued functions (Han and Kamber 2006).

For example it is possible to isolate three features for the single variable “initial menu selection.” If certain players tend to select options 1 and 3 more than average, it can be symptomatic of preference for solo play; while options 2, 4, 5 and 9 can point towards preference for social play; at the same time options 6, 7, 8 and 11, being non-interactive options dedicated to acquire extra information, might be highly represented in particularly dedicated players interested in extra information. “Solo Play”, “Social Play” and “Extra Info” are features extracted from the attributes of the variable “initial menu option selection” (see Table 13.2)

At this point it is possible to create a model to describe player behavior for the single variable “Initial Menu Selection.” Realizing that Solo Play and Social Play are mutually exclusive options helps considerably: in fact Solo Play and Social Play are the two opposite polarities of a single dimension describing one aspect of player behavior relative to “initial menu selection”. A second dimension is individuated by the feature ‘Extra Info’ that describes how much players tend to seek additional information and content. This second dimension is independent and perpendicular to the previous dimension. This model individuates four player types: Dedicated Hermits, Dedicated Party Animals, Light Hermits and Light Party Animals (see Fig. 13.1).

This is an example of a descriptive model created by classification. A predictive model could state that if certain conditions are met, players will tend to maintain a certain behavior. For example: if players show tendencies towards Solo Play within

Fig. 13.1 Model of player behavior for the variable “initial menu selection” in GoW2



the first week, there is a 80% chance that those players will maintain that behavior for at least 3 months. This model was created through a manual process of extracting and selecting features based on designers’ expert knowledge, which needs to be verified. This expert knowledge provides labels. A possible strategy for the process of creating models in this manner will be discussed in detail in Sect. 13.3. Alternative strategies for extracting and selecting features and deriving models based on automatic and algorithmic solutions, even without pre-existing labels, is discussed in Sect. 13.4 and Chap. 12. For more information on datamining approaches to models see Sect. 12.2.4.

13.2.5 From Variables to Models: A Process Along a Continuum

Summarizing the key concepts presented until now, it is possible to see how there are several progressive steps to move from the systems that compose a certain game, to models generated to describe the behavior of players interacting with those systems. It has also been demonstrated how there is not always a clear-cut division between those steps: the process of deriving descriptive and prescriptive models from the game systems, to a certain degree, is a continuum more than a succession of steps.

Figure 13.2 shows the process of defining variables from a game’s systems, measuring them and turning them into metrics data, extracting features and selecting those features with the intent of generating player models. All these actions are part of a continuum, where a game exists first as an independent object with unrealized potential (left hand side of the picture) and progressively, step by step, the actions of players are first measured and then interpreted into features, eventually generating models of player behavior (right hand side of the picture) actualizing the potentiality of the game.

The arrows on the top and bottom of the picture account for the fact that the process of defining variables and metrics can really benefit from human-driven sensitivity,

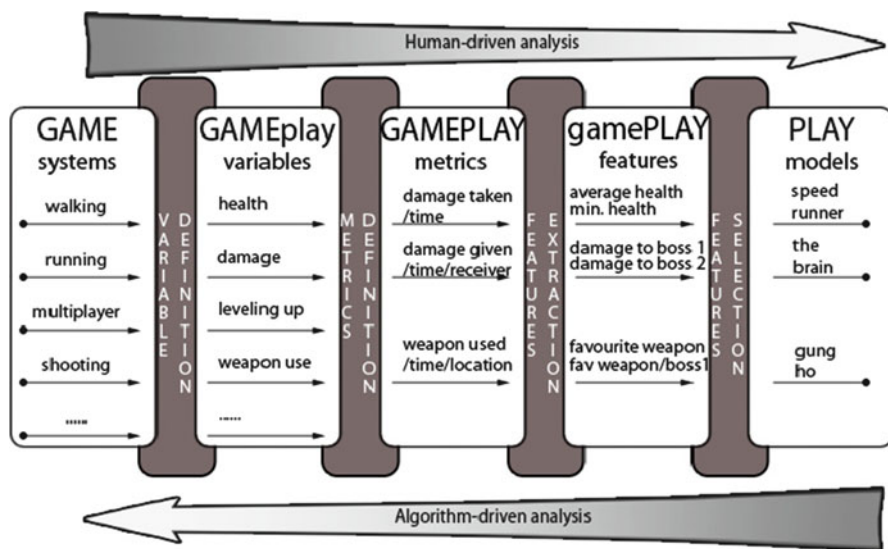


Fig. 13.2 The continuum where the game as a collection of systems becomes interpreted step by step through players actions

(what will be described in Sect. 13.3 as top-down, designer-driven approach), while the processes of feature extraction, feature selection and model training are more aptly tackled by algorithmic methods (referred in Sect. 13.4 as bottom-up algorithm-driven approach). This does not mean that human efforts should not be utilized in feature extraction or selection (for example to define labels), or that algorithms should not be used to select variables and ways to measure them, it simply points towards the fact that algorithms have proven successful in the later stages while human eye and domain knowledge has been so far irreplaceable in the first steps.

The figure attempts to capture the process in which a game, seen as a collection of systems, is analyzed to abstract those variables that more likely will provide answers to questions about the design. Those variables are then measured and features extracted from these measures, finally generating high-level information. For example, one of the most basic systems that usually exist in most games is a system that defines the state of the avatar as alive or dead. The variable that accounts for this system is “health”. Health is easily measured by monitoring the damage taken by the player controlled character. From this raw data, several features can be extracted, including “average health” during a whole play session or “minimum amount of health” reached in a specific location. Some of these features can be utilized to compare many players, and to look for similarities and differences and eventually create clusters or models of behavior.

The different steps proposed in order to move from systems to variables to metrics to features and eventually to models are strongly reminiscent of a framework proposed by Hunnicke et al. (2004) to formalize both the design and the consumption

of games. The Mechanics, Dynamics, and Aesthetics (MDA) framework breaks down the process into three components:

- Mechanics are the rules, the algorithms and data structures in a game
- Dynamics are mechanics actualized through player behavior and interacting with other mechanics
- Aesthetics are the hypothetical emotional responses evoked in the player during play.

Attempting to map these components onto the steps defined earlier, it is possible to say that mechanics encompass systems and variables since they represent data structures before any interaction with players; dynamics include metrics, telemetry and features since they can only exist with player interaction; and aesthetics maps perfectly onto models that describe and predict classes of player behavior, so much so that labels used to address player types, often closely resemble aesthetic values mentioned by the authors in the MDA framework: frustration – the challengers, fantasy – the escapist, fellowship – the socialite.

13.2.6 Game Telemetry, Metrics and Player Modeling

As we have seen, it is fairly straightforward to answer direct and specific questions about software stability or balancing the design by interrogating the data. The most complex task to achieve with the use of game data is exploring how players make use of a game, comparing the intents of the designers with the actual behavior of the players (Drachen and Canossa 2009; Drachen et al. 2009; Gagné et al. 2011, 2012).

Even if it is impossible to assess reasons and motivations behind players' actions just by looking at their in-game behavior, analysts still rely heavily on game metrics to evaluate user behavior. The most common approach to assess user behavior seems to be player profiling or player modeling. As expressed by Smith et al. "player modeling is a loose concept. It can equally apply to everything from a predictive model of player actions resulting from machine learning to a designer's description of a player's expected reactions in response to some piece of game content (Smith et al. 2011)."

Houlette first described models of individual players that are created by monitoring gameplay metrics (Houlette 2003). Charles and Black (2004) added explicit interrogation of players to the metrics-based inductive approach described by Houlette (2003). In both previous cases, player models are generated for greater and better adaptation of game system or shaping the behavior of bots (forward simulation and prediction of moves).

On the other hand there is also a wealth of research on player models used to imply players during design even before a working game prototype is produced (Canossa 2009; Schell 2008), these models are not computational but derived from designers' mindsets. In similar manner computational models have been created to verify designers' models and assumptions; these computational models have proved

invaluable tools (Drachen et al. 2009) also with the purpose of predicting player behavior (Mahlman et al. 2010).

Game designers often form non-explicit, internal models of players from a mixture of observation and accumulated design experience. These models have a considerable impact on level design, game design and production, influencing the gameplay experience at a far deeper level than any adaptive game system. In general, it seems possible to assert that player models help the process of interpreting or predicting actions and behaviors observed in players, independently whether they are physical individuals, bots or implied instances of hypothetical behaviors. Player models are sense-making lenses that allow the extraction of meaning from behaviors in games.

Player modeling consists of capturing behavioral data during the game and individuating groups of players with similar behavior or hypothetically similar behavior, if the model is theoretical and not based on empirical player data. Sections 13.3 and 13.4 will present two complementary strategies to select variables and create player models.

13.3 Designer-Driven Player Models: Play-Personas

The first approach described to derive player models from game behavior data is the play persona. Play personas are “[...] clusters of preferential interaction (what) and navigation (where) attitudes, temporally expressed (when), that coalesce around different kinds of inscribed affordances in the artifacts provided by game designers.” (Canossa and Drachen 2009a).

The play persona construct (Canossa and Drachen 2009b) is derived by combining goal-oriented design practices from human–computer interaction (Cooper 2004; Cooper et al. 2007; Pruitt and Grudin 2003) and theories from psychology of personality (Goldberg 2001; Costa and McCrae 1985).

13.3.1 HCI’s Personas

HCI’s contribution, the persona, is a detailed textual description of the hypothetical users for a certain product; it arose as a successful strategy to imply ideal users during the whole design process, as such the persona construct has been defined as “[...] an archetypical representation of real or potential users. It’s not a description of a real, single user or an average user. The persona represents patterns of users’ behavior, goals and motives, compiled in a fictional description of a single individual” (Blomkvist 2002). Cooper finds that these “archetypes that represent distinct groupings of behaviors, attitudes, aptitudes, goals, and motivations” (Cooper et al. 2007) greatly help developers understand the end user and to foresee its way of interacting with the product.

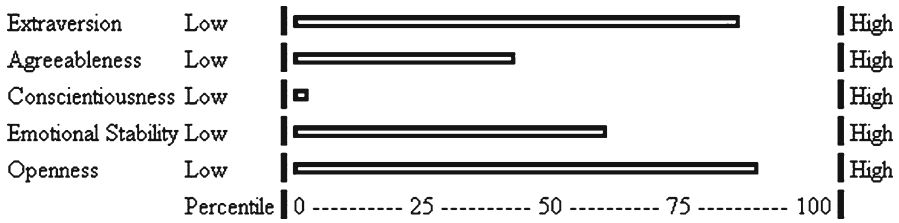


Fig. 13.3 A typical personality profile scored according to the IPIP NEO (International Personality Item Pool) Inventory (available at <http://www.personal.psu.edu/j5j/IPIP/>)

13.3.2 Psychology of Personality: The Five Factor Model

Theories from psychology of personality also greatly influenced the definition of play personas. Personality is defined as a collection of recognizable and reoccurring patterns of a rational, emotional and behavioral kind. Personality theories strive to understand and predict human behavior by observing people across various situations.

While type-based theories attempt to classify people into a number of categories (such as the 16 Myers-Briggs types), trait theories describe personality according to scores along certain dimensions. Trait theories do not attempt to fit profiles into pre-existing boxes, they attempt instead to describe a profile by scoring it according to certain parameters. The most accredited trait theory in personality studies today is the Five Factor Model (FFM) (Costa and McCrae 1992; Goldberg 1993). These factors are five broad domains or dimensions that are used to describe human personality. The framework represents a robust model for understanding the relationship between personality and various behaviors. The FFM was derived following the lexical hypothesis: “Those individual differences that are most salient and socially relevant in people’s lives will eventually become encoded into their language; the more important such a difference, the more likely is it to become expressed as a single word” (Allport 1936; John et al. 1988).

This hypothesis implies that studying how language encoded interpersonal differences can lead to an inclusive taxonomy of personality traits.

13.3.3 Play Persona as Synthesis of HCI Practices and Psychology of Personality

Differently from HCI’s persona, the play persona is not just limited to a narrative description of motivations, needs and desires distilled from ethnographic interviews, but instead it is represented as a set of scores along domains or dimensions, in the same manner that different personalities are described by the FFM (see Fig. 13.3).

In the Five Factor Models the domains were derived, according to the linguistic hypothesis, from a factor reduction of a large thesaurus of linguistic descriptors for personality. For games, there is no such thing as a shared universal language; hence the domains need to be defined ad hoc for each game under scrutiny. Similarly to the linguistic hypothesis, by examining all the actions that could be performed by players in a game, it is possible to capture all the behaviors that can theoretically be displayed by players in a certain game. The equivalents of linguistic descriptors in games are all the possible actions that players can perform and the resulting consequent changes to the game state. Hence, domains can be derived by grouping all the possible actions along the dimensions of play.

13.3.4 Variable Definition

The process of coalescing all possible actions and changes in the game state into coherent dimensions of play, corresponds to the first step towards the creation of models: the definition of variables. To exemplify this process we can look at a game like *Left4Dead* (L4D) (Valve Corporation, 2008) and how variables can be selected. The game is a first-person, cooperative shooter with a survival horror theme. Players are asked to survive hordes of infected enemies and escape the location to a safer area. Since players control a character, it is possible to apply the grid proposed by Tychsen and Canossa (2008) to classify different variables, namely *navigation*, *interaction*, *narrative* and *interface*.

13.3.4.1 Navigation Variables

The game presents several systems to allow navigation of the environments. Each system is identified as a variable. These variables are as follows:

- Walk
- Run
- Crouch
- Jump
- 180° spin

All of these variables have a time and location value; as such the context in which the actions are performed is valuable. Whether running takes place in brightly lit or dark spaces can be very informative in terms of prototypical player behavior.

13.3.4.2 Interaction Variables

The game offers a number of systems for players to interact with objects, characters and other players. Each system is identified as a variable. The events are as follows:

- Primary attack (ranged)
- Secondary attack (melee)
- Reload
- Use (context sensitive) (open and close doors, operate minigun, etc.)
- Damage to self/enemy/team mate (direct or indirect)
- Death to self/enemy/team mate (direct or indirect)
- Headshot to self/enemy/team mate
- Heal self/team mate
- Revive team mate
- Being revived by team mate
- Hand out first aid/pills
- Receive first aid/pills
- Partial/full reload
- Crouch-shooting (more precise)

As with the navigation variables, the context in which the actions are performed is very interesting, therefore both the time and the location of these actions is relevant.

13.3.4.3 Narrative Variables

The game offers four characters for players to select from. Although functionally equivalent, they provide a choice in terms of gender and ethnicity. A second narrative variable is the triggering of cinematics and other game events.

- Character selected initially.
- Triggering of different cut-scenes or level end cinematics.

Location is not really important for any of these variables; on the other hand time, although unimportant for character selection, plays an important role when examining the triggering of various cinematics.

13.3.4.4 Interface Variables

These variables map onto operations performed through the inventory or equipping items. They are as follows:

- Selecting any of these objects: flashlight, first aid kit, pain killers, Molotov bomb, pipe bomb
- Equipping weapons: pistol, dual pistol, sub machinegun, pump shotgun, auto shotgun, assault rifle, hunting rifle

As with navigation and interaction variables, the context in which the actions are performed is very interesting, so both the time and the location of these actions are relevant.

13.3.4.5 Aggregated Variables

The list of variables presented here can be grouped into several dimensions; guided by expert designers' knowledge, four domains have been chosen that point towards hypothetical superficial motivations for the actions performed. These four dimensions have not been derived through factor reduction, there is a clear interpretive bias operated any time humans are asked to group elements in categories, even if it is designers with exhaustive domain knowledge. The way variables have been grouped under these four domains is by no means the only possibility; on the other hand, nearly all instances of player behavior in L4D can be captured by this hermeneutic grid.

Surviving

Self-preservation is paramount, it informs everything from navigation attitudes to weapon of choice. All actions have the sole purpose of preserving hit points and avoiding damage. Variables that directly relate to this domain are: players using health packs and pain killers on themselves, player health level, weapon selection, attack type (ranged or melee), preferred environment type (dark or bright).

Killing

It is always preferable to dispatch few infected than preserving own health; the sense of triumph when dominating the opposition is a main driver for action, players are polarized according to their ability or desire to dispatch enemies. Variables that directly relate to this domain are: headshots inflicted on infected, accuracy of shots, number of infected dispatched, proficiency of use of pipe bombs and molotovs.

Helping

This dimension shows players' attentiveness to each other and awareness of the whole group. Actions are not necessarily performed out of selflessness, but because of the thrill of the fray. The main variables underlying this dimension is the ability of players to detect when other players are under attack by responding to emergency signals and killing infected and special infected that are dealing damage to team mates, finally the act of reviving team mates is the coronation of their efforts.

Healing

Healing is when players sacrifice their own potential wellbeing for the good of the group. It represents an extreme version of helping but it is different enough to

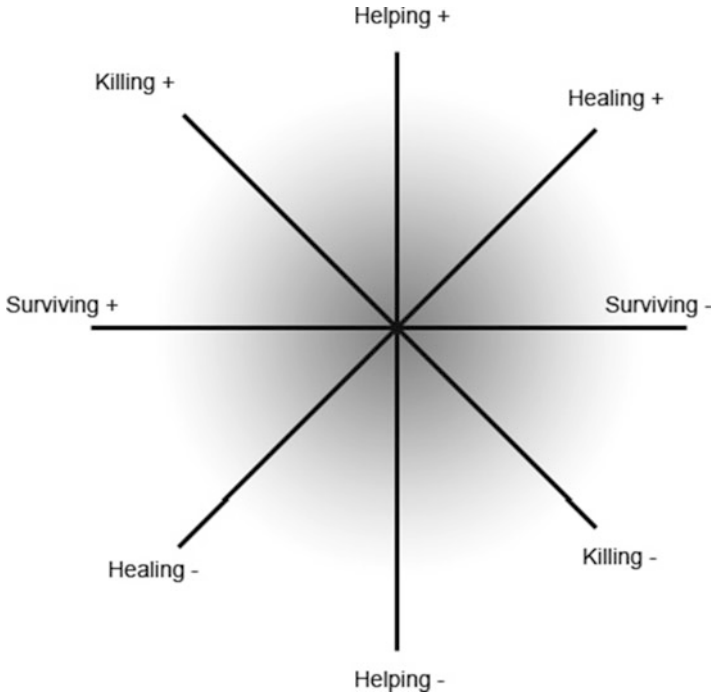


Fig. 13.4 The 4-dimensional possibility field

warrant the existence of a different domain. Giving health packs or pain killers to other players is the main variable underlying this behavioral axis.

13.3.5 Metrics Definition

These four dimensions form the axes of a 4-dimensional space that attempts to contain all the possible players' behaviors in the game (Fig. 13.4). At the same time these domains help selecting which game variables should be monitored, and how, in order to gather features for creating players' profiles of behavior.

13.3.5.1 Surviving

In order to measure the variables listed earlier, logs with location and time stamps should be created for each time players use health packs and pain killers on themselves, or each time a player receives damage, or each time players deal damage, including information about the type of weapon or melee tool and the entity that

receives or generates player damage. All the metrics listed until now are triggered, but in order to account for the preferred space, it is necessary to log sampled information about the coordinates of the character, a suggestion is to sample location every second and interpolate positions in between.

13.3.5.2 Killing

Measuring actions related to killing behavior is accomplished by recording each bullet fired, each infected injured or killed, each headshot and each pipe bomb or molotov used. Each one of these triggered logs should include location and time. Even for this behavior, it's relevant to track sampled information about the coordinates of the character and the infected that was hit.

13.3.5.3 Helping

It is not easy to infer whether a player is aware of other teammates' troubles, no change in the game state can account for that, but it is possible to log each time a player shoots an infested that is dealing damage to a team mate. This is accomplished triangulating the metric "player receiving damage" defined under "surviving" domain and "player injuring or killing infected". This triangulation should also capture the event of a player being under attack by a special infected, such as hunters or smokers. Aggregating variables in such manner already generates features ready for creating models. There is another state where players can be incapacitated (but not dead) and they need to be revived; this event needs to be recorded together with time and location. Rescuing survivors after a death event, causing a respawn event, is the last metric relevant to this domain.

13.3.5.4 Healing

Each time a player gives health packs or pain killers to other players should be recorded with time and location. It is not necessary to monitor all variables in order to create models to account for all possible behaviors.

13.3.6 Feature Extraction

As we have already seen in Sect. 13.2.3, sometimes it is not sufficient to rely on simple metrics to account for many of the behaviors that are necessary to score players along the four domains. For example, "accuracy" is not a simple variable that can be measured, but it requires triangulation of "number of bullets fired"

with “number of times players dealt damage”. In general it is possible to say that a feature emerges any time more than one metric is triangulated, as seen in the example of the “helping” dimension.

13.3.7 Feature Selection and Profile Creation

Once all the features have been extracted from the game telemetry data, a certain number of them are necessary to generate informative models. With a human-driven method, the necessary features are already selected in the previous steps, and this can generate truisms, but as we will see later, for a computational approach, this is one of the most complex tasks. Comparing the score of each player with the baseline for that dimension, allows for classification of players according to the 16 types that emerged as relations between the four domains initially individuated, as it is shown in Table 13.3.

Players are scored as “high” or “low” for each of the domains individuated according to whether they exceed the value set by the baseline or not. Keeping these values down to two, although losing granularity and resolution, simplifies immensely the possibility space. Unfolding the possibility space consists of exploding all the combinations of scores along domains as shown in Table 13.3 (Canossa and Drachen 2009b).

The models of player behavior emerging from play personas can be described as in-game personality profiles derived from player actions. In fact, adopting a play persona approach, it is possible to deduce in-game personality dimensions from any game’s core actions and use these dimensions to describe the behavior of players and inform the generation of archetypical player’s profiles (Canossa and Drachen 2009a).

When defining the possible variables to create models of player behavior it is important to ascertain whether these models should try to capture the most common traits encountered or whether instead it is considered more interesting examining extreme patterns of behavior and capture fringe actions. The immediate implication of this choice is seen in how statistical outliers are treated: if models attempt to capture normal attitudes, then outliers are discarded in favor of more statistically relevant data. On the other end, if the focus of the model is to describe extreme possible patterns of actions afforded by a certain game, then the outliers become pivotal to create archetypes of aberration.

For example, consider the practice of speed running: it consists of playing a game, or part thereof, with the intention of completing it as fast as possible. There are statistically few players completing games in such a way, but creating a model also encompassing this behavior can be extremely informative for game developers, it could in fact answer questions such as “what is the fastest time a certain level can be completed in?”, “what is the minimum set of actions necessary to finish the game?” or “is there a dominant winning strategy?”. Normally speed runners generate data that falls on the fringe of the usual Gaussian distribution curve, as such

Table 13.3 Example: A player scoring less than average on “survival” but goes the extra mile to face more infected than the average, while not helping or healing team mates will be classified as a “Grunt”

	Fugitive	Grunt	Samaritan	Doctor	Rambo	Red Cross	Expert	Rookie	H1	H2	H3	H4	H5	H6	H7	H8
Survive	+	-	-	-	+	-	+	-	+	-	+	-	+	+	+	-
Kill	-	+	-	-	+	-	+	-	-	+	-	+	+	+	-	+
Help	-	-	+	-	-	+	+	-	-	+	+	-	+	-	+	+
Heal	-	-	-	+	-	+	+	-	+	-	-	+	-	+	+	+

would be treated either as flukes or as mistakes and dropped. In normally distributed datasets, nearly all observations lie within three standard deviations of the mean value, speed runners are known to be even four or five standard deviations from the mean value. Independently of the small population represented by models based on these values, it can be of interest to examine these extreme behaviors.

On the other hand, if the purpose of the model is to predict probable, likely player behavior, then it becomes necessary to make use of robust measures such as the median value rather than the mean.

More polarized play personas are the most interesting ones to flesh out, since the extremes can illustrate different play-styles within the game more clearly. However, actual physical players more often will fall into one of the less interesting hybrid categories, given that they could easily switch between extreme approaches, as shown by Canossa et al. (2009a). Nevertheless modeling extreme behaviors can trace the edges of the possibility space; for this case study the six labeled profiles represent the polarized profiles while the hybrids are more typical profiles able to capture behavior of the large majority of players.

13.3.8 General Considerations on Play Personas

The power of this approach is the analytical framework that allows the definition of measures for all variables identified. As stated earlier in Sect. 13.2.5, it is precisely at the beginning of the process that a human-centric approach can bring the most benefits. Furthermore; play personas can be utilized a-priori, even before the game-play is fully implemented, as a metaphor that enables designers to create content and spaces around behavioral profiles before any real players had the chance to interact with the game. Not only can this approach help define models of player behavior as reflective lenses to uncover whether or not players actually embody intended play-styles, but it can be instrumental in exploring hypothetical behavioral patterns during the early stages of the design as evocative metaphors since a number of indicative play personas can be created around the central gameplay mechanics, and used by the level designers to accommodate the different play-styles within the virtual environments. It is in this respect that play-personas are both theoretical models of ideal users (metaphors) and data-driven representations of player behaviors (lenses).

The four dimensions applied here are only an example of a hermeneutic grid that can be used to filter gameplay variables and interpret behaviors. Similar work has been conducted by Wong et al. (2009) applying a grid derived from Bartle's work (Bartle 1996). Charles et al. (2005) have instead shown how Campbell's hero's journey can be utilized to infer such a grid. The power of the play persona approach relies in the fact that no a-priori theory exists, but instead a grid is derived ad hoc from the affordances provided by each game.

The play persona approach is only one of the many possible strategies that can help defining variables and measures of gameplay behavior in order to produce

player models. The last part of the chapter will briefly explore and summarize computational strategies to accomplish the same purpose: creating models of player behavior starting from traces left by players interacting with games. For a more detailed account on the subject the reader is recommended to see Chap. 12.

13.4 Algorithm-Driven Player Models: Computational Methods

Algorithmic approaches for creating models of player behavior in games have been employed for a long time starting with Houlette's model (2003). Smith et al. (2011) provided a good overview of existing computational methods. The authors analyzed how different player models are derived and classified the models as Empirical or Theoretical. Empirical models (further divided as Induced and Interpreted) are based on actions of real human players. Theoretical models (further divided as Analytic and Synthetic) are derived without any direct references to data. They also propose another way to subdivide modeling techniques: subjective models (Interpreted and Synthetic) are based on the credibility of the analyst or designer that defined them, while objective models (Induced and Analytic) are based solely on the intrinsic logic of the methods employed. Models derived using manual approaches, such as the play persona, are definitely subjective but can be both empirical and theoretical according to whether play personas are developed as a-priori metaphors or a-posteriori lenses. Smith et al. introduced four basic categories of player models: induced, interpreted, analytic and synthetic. Computational models are mostly induced and analytical, while models derived manually and based on expert knowledge are frequently interpreted and synthetic.

13.4.1 *Induced*

Induced models are based on game telemetry and are derived through objective inductive analysis. Often these models are created utilizing machine learning or other statistical analyses applied to features that are already extracted from metrics data defined by human analysts. Play persona models are induced by human reasoning.

13.4.2 *Interpreted*

Interpreted models are subjective since human interpretation is needed to map the empirical observations to informative descriptions, for example Bartle's four player types (Bartle 1996). Human experience and domain knowledge are prerequisites for successful interpretation.

13.4.3 Analytic

Analytic methods use automated methods to extract the truths inherent in a game's design in a useful form. Normally, the rules of a game are exhaustively searched to produce the best outcome to describe a variety of player behaviors. This approach requires describing all the possible game states in a way that is computationally understandable to be used as rules or constraints for a search algorithm. Analytical models do not reference any particular framework or theory; the naked rules of a game directly define an implicit model, which describes players as potentially capable of doing anything the rules allow.

13.4.4 Synthetic

Synthetic player models are founded on beliefs or assumptions generated outside the game itself. For example, the application of Bartle's four types to another game that is not MUD is a synthetic model. Synthetic models are often transferred interpreted models derived from another game or codified hunches, intuitions, assumptions about the audience and other beliefs which are not traceable to any particular piece of evidence.

13.4.5 An Overview of Automated Approaches to Creation of Models

Traditionally, automated approaches to model creation are applied to existing datasets, relational databases or data warehouses, meaning that the process of analyzing game systems, defining variables and establishing measures for such variables, falls outside of the domain (Han and Kamber 2006). Therefore, the phases that we will be examining are only the last three:

- Feature extraction: the process of defining or creating a feature
- Feature selection: the process of selecting the necessary amount of features that result in the models with the best fitness
- Model training: the process of creating models

13.4.5.1 Feature Extraction or Construction

Feature extraction is the process of reducing the dimensionality of the dataset, defining the form of the information to use in advanced data mining practices such as creating models. Feature extraction practices, also sometimes addressed

as feature construction, are often considered forms of data pre-processing. The purpose of feature construction is to construct a single feature representing different attributes. This often leads to feature selection or feature aggregation.

Common tools in feature extraction are:

- descriptive data summarization with statistical methods such as mean, median and mode,
- frequent pattern mining; frequent sequence mining is also utilized in the process of model training,
- automated signal processing algorithms such as fast Fourier transformations, used to move from time domain to frequency domain, mostly used to process psychophysiological datasets such as beat detection in heart rate signals.

13.4.5.2 Feature Selection

In feature selection the purpose is to individuate only on the most relevant and the most discriminating features. A similar process is feature aggregation: this process takes a varied set of values and returns a summary of it. This summary may be, for instance, the mode – the most frequently occurring value; the mean value of the set, if the values are numerical; or the median or “middle” value, if the values are ordered (Han and Kamber 2006).

Automated feature selection relies on algorithms to search the attribute space and drop features that are highly correlated to others; these algorithms can be as complex as sequential forward selection and neuro-evolutionary preference learning (Martinez and Yannakakis 2010) but also fast clustering algorithms, such as k-means, can be applied to find the most relevant features since the presence of features that are not relevant for the definition of types affects the similarity measure degrading the quality of the clusters found by the algorithm.

Utilizing clustering algorithms to remove features begins to blur the process of feature selection into model creation.

13.4.5.3 Models

As seen in Sect. 13.2.4, models afford better understanding of a dataset; in this particular case they can summarize the behavior of millions of players. Models can be both descriptive interpretations and predictive statements. Models are first created, then trained with data and finally their prediction value is tested.

Machine learning approaches are often used to create and train models. In machine learning there are two main methods to derive a model from a dataset: supervised and unsupervised learning. In supervised learning, a complete set of labeled data is available; in unsupervised learning no labels are known. For in depth look at data mining models, the reader is strongly recommended to read Chap. 12.

13.5 Conclusions

In games, as for any other form of analysis from business intelligence to web analytics, it is a lot easier to address specific concrete questions. If developers want to know where players spend most of their time in a certain level or what is their favorite weapon, it is a fairly straightforward process to isolate relevant game variables, measure them, select and extract features to answer precisely that question. On the other hand, performing exploratory analyses looking for trends, patterns and clusters of similarities requires much more subtle and complex approaches. In this chapter, two main alternatives were shown: a top-down method, where features are selected according to experts domain knowledge and then models are hypothesized based on such features; and a bottom-up method, where clusters based on similarities are generated computationally and models are based on them. The top-down approach has its biggest limit in the fact that it can potentially lead to tautologies and truisms because of the fact that certain features are deemed essential by expert designers. It is possible to generate models in such a way that they cannot disprove initial hypotheses or suggest alternative features that better describe player behavior and in turn lead to better models. The bottom-up approach, here merely drafted but described in detail in Chap. 12, may appear more objective and falsifiable, but it contains a wealth of hidden bias. In fact, as shown by Thureau and Drachen (2011) different algorithms for classifying player behavior can lead to very different findings.

Both the a-priori top-down approaches and a-posteriori bottom-up approaches are biased by the hermeneutics of player behavior, and such a bias might be unavoidable. In fact it is the very same bias that enters the picture every time a continuous and heterogeneous stream of information – such as player behavior – is ordered, categorized and measured. Human behavior emerges as the response of an organism to its ecosystem, it possesses no intrinsic meaning; rather it needs to be interpreted. Both measuring and hypothesizing behavior is arbitrary to a certain extent; attempting to read meaning in the measures of such behavior is even more arbitrary.

Both top-down, expert-driven and bottom-up, algorithm-driven methods imply a certain degree of uncertainty – measurements and hypotheses destroy part of the knowledge of the system depending on which measurement are to be performed or which features are chosen to be prioritized.

Feature selection for player modeling is still a soft science; it needs to be tackled as trial and error where the experience and domain knowledge of a designer provides initial hypotheses to run the first algorithmic analyses and the computational models generated can supplement the original intuition of designers. The two approaches here described are in no measure exclusive, they are instead supposed to support each other, embracing the unavoidable bias and working around it, providing hermeneutic scaffolds to guide the interpretation efforts.

About the Author

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and The Royal Danish Academy of Fine Arts, School of Architecture. His doctoral research was carried out in collaboration with Io Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches: prototypical player behaviors are described procedurally and from those profiles game environments emerge that are able to accommodate all of the possibilities for action. In the course of his research, Dr. Canossa has published more than 30 articles, book chapters, and journal contributions, he has presented at several international conferences including Future Play, GDC San Francisco, GDC Canada, Mindtrek, IFIP Interact, IEEE Conference on Computational Intelligence and Games, ACM Foundations of Digital Games and DiGRA discussing topics from game user research, HCI, game metrics analysis and player experience. He has also received a Best Paper award at the largest media conference in Northern Europe, MindTrek, in 2009. Beside his academic activities he still maintain contact with the industry: his work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he carries on an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio, where he has worked on titles such as Kane & Lynch: Dead Men, Tomb Raider: Underworld and Kane & Lynch: Dog Days.

References

- Allport, G. (1936). *Personality: A psychological interpretation*. New York: Holt, Rinehart, & Winston publishers.
- Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*, 1(1), 19.
- Blomkvist, S. (2002). Persona – An overview. In *Theoretical perspectives in human-computer interaction*. Stockholm, IPLab, KTH.
- Brudvig, E. (2010). Halo: Reach post-beta interview. IGN. Retrieved June 6, 2010.
- Canossa, A. (2009). Play persona: Modeling player behaviour in computer games. The Royal Danish Academy of Fine Arts, School of Design/IO Interactive. Ph.D. thesis, Supervisor: Dr. Troels Degn Johansson, Denmark.
- Canossa, A., & Drachen, A. (2009a). Patterns of play: Play-personas in user-centered game development. In *Proceedings of DIGRA 2009*. London, United Kingdom:DiGRA Publishers.
- Canossa, A., & Drachen, A. (2009b). Play-personas: Behaviors and belief systems in user-centered game design. In *Proceedings of INTERACT 2009* (LNCS vol. 5727, pp. 510–523). Uppsala, Sweden: Springer.
- Charles, D., & Black, M. (2004, November). Dynamic player modeling: A framework for player-centered digital games. In *Proceedings of the international conference on computer games: Artificial intelligence, design and education* (pp. 29–35).

- Charles, D., McNeill, M., Mcalister, M., Black, M., Moore, A., Stringer, K. Kücklich, J., & Kerr, A. (2005). Player-centred game design: Player modelling and adaptive digital games. In *Digital games research association 2005 conference: Changing views – Worlds in play*, Vancouver.
- Cooper, A. (2004). *The inmates are ruling the asylum*. Indianapolis: Sams Publishing.
- Cooper, A., Reimann, R., & Cronin, D. (2007). *About face 3: The essentials of interaction design*. Indianapolis: Wiley.
- Costa, P. T., & McCrae, R. R. (1985). *The NEO personality inventory manual*. Odessa: Psychological Assessment Resources.
- Costa, P. T., & McCrae, R. R. (1992). *Revised NEO personality inventory (NEO-PI-R) and NEO five-factor inventory (NEO-FFI) manual*. Odessa: Psychological Assessment Resources.
- Drachen, A., & Canossa, A. (2009). Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th MindTrek*. Tampere: ACM-SIGCHI Publishers.
- Drachen, A., Canossa, A., & Yannakakis, G. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the IEEE computational intelligence in games* (pp. 1–8). Milan: IEEE Publishers.
- Edmunds, A., & Morris, A. (2000). The problem of information overload in business organisations: a review of the literature. *International Journal of Information Management*, 20(1), 17–28. ISSN:0268–4012, doi:10.1016/S0268-4012(99)00051-1, URL: <http://www.sciencedirect.com/science/article/pii/S0268401299000511>
- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2011). A deeper look at the use of telemetry for analysis of player behavior in RTS games. *Entertainment Computing–ICEC 2011*, 6972, 247–257.
- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2012). Analysis of telemetry data from a real-time strategy game: A case study. *ACM Computers in Entertainment*, 10 (3), Article No. 2. doi:10.1145/2381876.2381878, <http://doi.acm.org/>
- Goldberg, L. R. (1993). The structure of phenotypic personality traits. *American Psychologist*, 48, 26–34.
- Goldberg, L. R. (2001). Personality processes and individual differences – An alternative description of personality: The big-five factor structure. In S. E. Hyman (Ed.), *The science of mental health*. New York: Routledge.
- Han, J., & Kamber, M. (2006). *Data mining: Concepts and techniques* (Second Editionth ed.). San Francisco: Morgan Kaufman Publishers.
- Houlette, R. (2003, December). Player modeling for adaptive games. In S. Rabin (Ed.), *AI game programming wisdom 2*. Hingham: Charles River Media.
- Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A formal approach to game design and game research. In *Proceedings of the challenges in game AI workshop, 19th national conference on artificial intelligence AAAI'04*. San Jose: AAAI Press.
- John, O. P., Angleitner, A., & Ostendorf, F. (1988). The lexical approach to personality: A historical review of trait taxonomic research. *European Journal of Personality*, 2(3), 171–203. Wiley, Universität Bielefeld, Federal Republic of Germany.
- Mahlman, T., Drachen, A., Canossa, A., Togelius, J., & Yannakakis, G. N. (2010). Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 2010 IEEE conference on computational intelligence in games* (pp. 178–185). Copenhagen: IEEE Publishers.
- Martínez, H. P., & Yannakakis, G. N. (2010, October). Genetic search feature selection for affective modeling: A case study on reported preferences. In *Proceedings of the 3rd international workshop on Affective interaction in natural environments* (pp. 15–20). New York: ACM.
- Pincus, M., & Gordon, B. (2009). Ghetto testing and minimum viable products. Lecture at Stanford Technology Ventures Program, Stanford University.
- Pruitt, J., & Grudin, J. (2003, June). Personas: Practice and theory. In *Proceedings of the 2003 conference on designing for user experiences* (pp. 1–15). San Francisco: ACM.
- Salen, K., & Zimmerman, E. (2003). *Rules of play: Game design fundamentals*. Cambridge: The MIT Press.
- Schell, J. (2008, August). *The art of game design: A book of lenses* (p. 99). Amsterdam: Morgan Kaufmann.

- Smith, A. M., Lewis, C., Hullett, K., Smith, G., & Sullivan, A. (2011, July). An inclusive view of player modeling. *6th international conference on Foundations of Digital Games (FDG 2011)*, Bordeaux, France.
- Sydney Morning Herald (2010, May 25). Millions reach for 'Halo'. *Sydney Morning Herald*. Retrieved June 6, 2010.
- Thureau, C., & Drachen, A. (2011). Introducing archetypal analysis for player classification in games. *EPEX Workshop in FDG 2011*, Bordeaux.
- Tychsen, A., & Canossa, A. (2008). Defining personas in games using metrics. In *Proceedings of FUTURE PLAY 2008* (pp. 73–80). Toronto: ACM publishers.
- Wittgenstein, L. (2001). *Tractatus logico-philosophicus* (2nd ed.). London: Routledge.
- Wong, C., Kim, J., Han, E., & Jung, K. (2009). Human-centered modeling for style-based adaptive games. *Journal of Zhejiang University – SCIENCE A*, 10(4), 530–534.

Chapter 14

Gameplay Metrics in Game User Research: Examples from the Trenches

Anders Drachen, Alessandro Canossa, and Janus Rau Møller Sørensen

Take Away Points:

1. An overview of the fundamental approaches to working with behavioral metrics.
2. Four case studies of gameplay analyses from the development of the games Tomb Raider: Underworld and Kane & Lynch: Dog Days.
3. Recommendations and considerations on the application of metrics to game user research.

A. Drachen, Ph.D. (✉)

PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark

Game Analytics, Copenhagen, Denmark

e-mail: andersdrachen@gmail.com

A. Canossa, Ph.D.

College of Arts, Media and Design, Northeastern University,

Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,

Copenhagen, Denmark

e-mail: a.canossa@neu.edu

J.R.M. Sørensen

Crystal Dynamics/Square Enix, Redwood City, CA, USA

e-mail: januss@crystald.com

14.1 Introduction

Games User Research (GUR) has gained a lot of ground recently in game development. It has become apparent just how beneficial knowing your target audience really is. More and more, studios and publishers integrate user-oriented methods in their development process and report direct benefits from this practice, above and beyond simple focus group feedback. This because GUR is a field that is all about the user: their experience, their game, their feeling of fun, etc. As users are also customers, it has become a widely accepted notion that GUR helps to improve games and thereby sales, so the investment and growth in this area is currently considerable (Pagulayan et al. 2003; Pagulayan and Keeker 2007; Isbister and Schaffer 2008; Kim et al. 2008; Drachen and Canossa 2009a, b; Lewis-Ewans 2012).

One of the newer developments in the game development in general, as of the time of writing, is the increased use of game metrics to support development (Kennerly 2003; Thompson 2007; Kim et al. 2008; Drachen and Canossa 2011), inspired by the application in e.g. web analytics (Sterne 2002). Quantitative measures of players, processes and performance have a long history in game development and software development in general, but it is only in the past roughly 5 years that the application of metrics, calculated from client telemetry data, have become more mainstream to game user research and game development.

From the perspective of GUR, which is the focus in this chapter, the unique value of game metrics lies in the objective tracking of user behavior and subsequent translation into data that can be quantified and manipulated – and jointly analyzed with other sources of user data, e.g. from usability- and playtesting. Using telemetry, it is possible to evaluate designs and debug user experiences to a degree of detail that for example observational methods does not allow.

Game metrics can be obtained from a variety of sources, but the most common, and the one we focus on here, is telemetry. Game telemetry data are the raw units of data that are derived from e.g. an installed game client. Code embedded in the game client transmits data to a collection server about how a player interacts with the game; or alternatively telemetry data are collected from game servers (as used in e.g. online multi-player games) (Derosa 2007; Kim et al. 2008; Canossa and Drachen 2009; King and Chen 2009).

The challenge with the goldmine of data that game metrics comprise is to know what to look for, why to look for it and how to make it valuable for different stakeholders, so it becomes more than a basic tool but a basis for analysis. While there are some gameplay metrics that are universally useful to track and analyze, irrespective of the game – such as playtime, player progress through a games' levels, player ID, asset use etc. – the choice of what to track and how to track it is highly varied across games and specific features such as whether or not the game is persistent, single- or multiplayer, etc. Likewise, the design and research ecology that the metrics analysis becomes a part of differs from publisher to publisher and studio to studio. This makes it hard to generalize specific solutions across publishers, studios, and games, and it poses the inevitable question: What should we track, and how should we work with the data?

In this chapter, we will outline a few case studies from work at IO Interactive and Crystal Dynamics, both subsidiary studios of Square Enix Europe, showing some of the specific ways in which we have utilized gameplay metrics in practice, in the specific context of GUR (e.g. evaluation, playtesting, usability testing, and so forth). We try to illustrate how gameplay metrics are a useful source of data on player behavior during the development process, focusing on the games *Tomb Raider: Underworld* (2008, Eidos) and *Kane & Lynch: Dog Days* (2010, Eidos/Square Enix), as well as the multi-player form: *Fragile Alliance 2*. The cases presented are drawn partly from earlier research publications (Drachen and Canossa 2009a, b, 2011). We focus on methodologically simple, straight-forward analyses that do not require advanced statistical or data mining expertise (for more on game data mining see Chap. 12). The cases are all from the third-person action adventure- and shooter genres, but many of the ideas can potentially be applied across game forms.

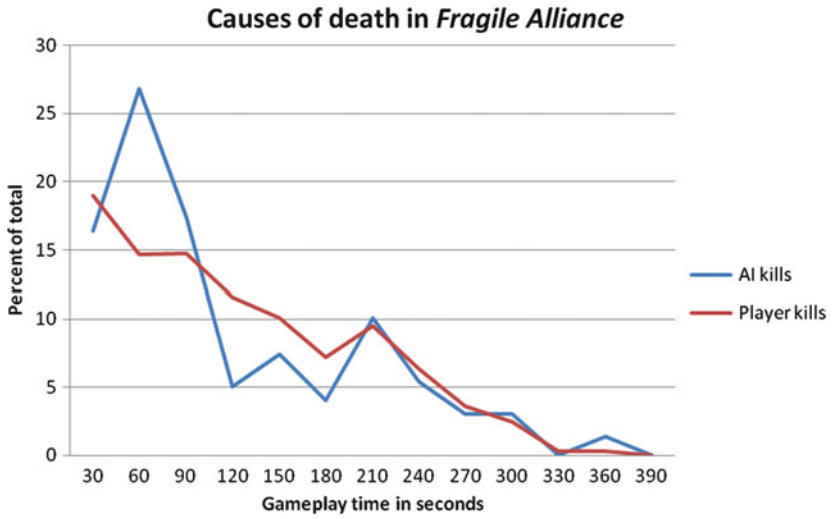
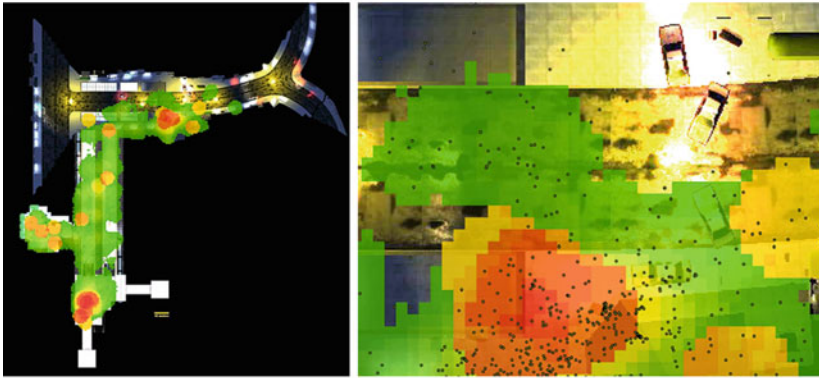
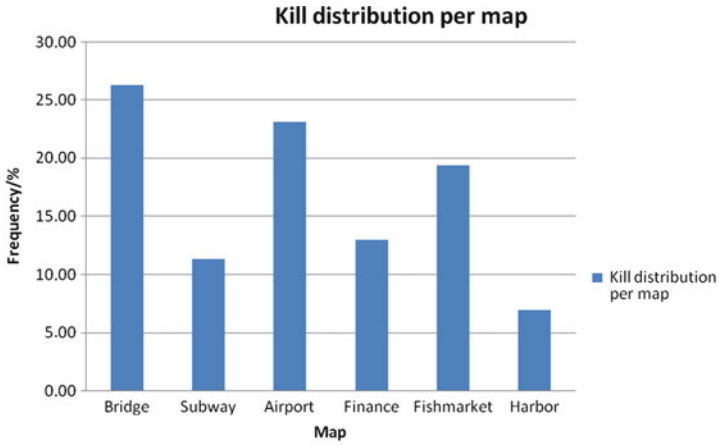
We will not be arguing for the superiority of game metrics above other sources of user-oriented data or methods in GUR and user-oriented testing, nor are we asserting that data mining has more value than other user research methods. In fact, we will describe how the practice of metrics analysis often evolves interplay with other research methods and the context of the research (the game development process). Towards this aim, we will focus on how the designers and GUR people at IO Interactive and Crystal Dynamics employ game metrics analysis with other sources of user-oriented data. This topic is also discussed in Chaps. 21 and 22 of this book.

14.2 Gameplay Metrics: Capturing User Behavior

Game metrics are discussed in more depth in Chap. 2, but it is worth mentioning that we view them as a specific form of business intelligence data, derived from the people, customers and processes involved in the business of games. Game metrics present the same potential advantages as other sources of business intelligence, i.e. support for decision-making in companies, at all levels of a company and practice. As will all other forms of business intelligence data that we use to develop and research games, the fundamental way to work with game metrics is through analysis (Vercellis 2009).

When game metrics are employed in Game User Research, the type of metrics we work with are generally referred to as **gameplay metrics**. These are measures of player behavior, e.g. navigation, item- and ability use, jumping, trading, running and whatever else players actually do inside the virtual environment of a game (whether 2D or 3D).

Player behavior measures can be anything happening inside the game environment, and gameplay metrics can be viewed as the “breadcrumbs”, the tracks, left by players in the games. Any action the player undertakes while playing can be tracked: every time a door is opened, a gun is fired, a treasure uncovered or a level completed, a telemetry tracking system can note down where and when that action happened (see Figs. 14.1 and 14.2 for some examples). Gameplay metrics are the most important form of game telemetry when the purpose is to evaluate design and user



experience, and are furthest from the traditional perspective of the revenue chain in game development, and hence are generally under-prioritized (Pagulayan et al. 2003; Pagulayan and Keeker 2007; Isbister and Schaffer 2008; Kim et al. 2008; Drachen and Canossa 2009a, b). However, analysis of gameplay metrics provide the opportunity to address key questions, including whether any game world areas are over- or underused, if players utilize game features as intended, or whether there are any barriers hindering player progression. This kind of instrumentation data can be recorded during all phases of game development, as well as following a launch.

As a data source, gameplay metrics – and the methods used to obtain knowledge from them, such as data mining, machine learning and statistical analysis (Chap. 12) – supplement other types of user-oriented data and the methods used to analyze them, e.g. *usability evaluation* (measuring ease of use of the game) and *playability* and *user experience evaluation* (other methods exploring if players have a good experience playing the game), by offering insights into how people are actually playing the games being studied – i.e. their behavior – in detail. This has led numerous developers and publishers to combine gameplay metrics analysis with other sources of information on the player experience, for example questionnaires, interviews and gameplay observations and recordings (also discussed in Chaps. 21 and 22, see also Isbister and Schaffer 2008; Kim et al. 2008; Seif El-Nasr and Zammitto 2010; Lameman et al. 2010; Lewis-Ewans 2012).

14.3 Approaches to Working with Gameplay Metrics

When examining the available literature on game analytics from the different areas of the game industry, as is evident in the various chapters in this book (e.g. Chaps. 4, 7, 12, 17, 18, 19, 21, and 22) as well as various conference presentations (Zoeller 2011) and articles and books (Kim et al. 2008; Isbister et al. 2008; Drachen et al. 2009; Lewis-Ewans 2012), and the older and much more developed fields of web analytics and Business Intelligence (e.g. Sterne 2002; Vercellis 2009), it becomes apparent that there are different ways to approach the kind of detailed user behavior work that gameplay metrics permit.



Fig. 14.1 Examples of various non-spatial and spatial syntheses of gameplay metrics data from *Fragile Alliance 2*, capturing various features of user behavior. Note that the visualizations were generated during development, so the relative distributions of event frequencies and map designs are different in the launched version of the game: (top) Kill distribution (in percent) across six maps from the team multi-player shooter *Fragile Alliance 2*, comparing lethality. (middle left): Heat map (developed using a density kernel function) from the Subway map of the same game, and a (middle right) closeup from the top of the map, showing the pinpoint spatial accuracy of individual death events. (bottom) A graph showing the overall causes of death for a group of playtests on the Subway map, indicating a strong AI influence in the first 60 s of the game at that point during production – in line with the designers intention

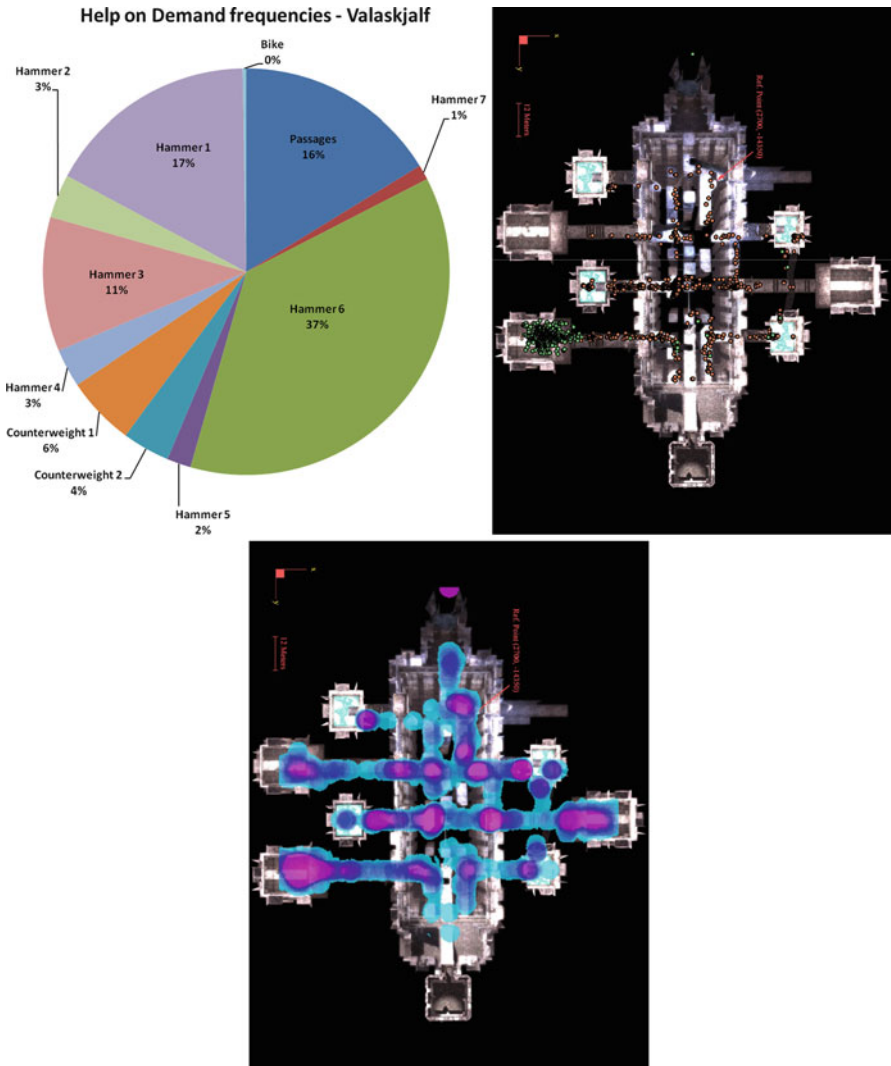


Fig. 14.2 Examples of various non-spatial and spatial syntheses of gameplay metrics data from *Tomb Raider: Underworld*, capturing various features of user behavior. (top left) pie chart showing the distribution of help requests for the different puzzles or challenges in the Valaskjalf map unit. (top right) A map of the position of players in the Valaskjalf map unit of *Tomb Raider: Underworld*, when they request help for two of the puzzles in the map unit (brown and green points respectively) (the game features a built-in help system to assist players with puzzle solutions). (bottom) a heat map of the player death events (purple most events, light blue fewest)

Fundamentally, game analytics – despite the term – is either performed via **analysis** or by **synthesis**, both classic scientific methods. The difference can be subtle in practice, but basically *analysis* is when we break down a complex whole into parts or components, whereas *synthesis* is the opposite procedure, i.e. combining

separate elements or components in order to form a coherent and complex whole. For example, breaking down behavioral data into smaller parts (e.g. time spent per checkpoint, number of button presses) to gain a better understanding of the individual components is analysis, while a chart showing the number of daily active users is a synthesis of individual components (time, number of users etc.). As such, the two methods go hand in hand and complement each other.

Both analysis and synthesis can be initiated by fairly open-ended or specific questions, roughly correlating with the concepts of **explorative** vs. **hypothesis** driven research from scientific theory (see also Chap. 12). What this means is that the approaches we use to find the answers to questions are either of a type where we are looking to confirm some idea we have and are looking for confirmation, or alternatively have a pretty good idea about the possible answers (hypothesis-driven); or alternatively more open, where we are not sure what the answer to a given question is, or have a hard time predicting the possible answers.

Explorative metrics research is when the possible answers cannot or are hard to predict from looking at the game design. For example, in a free-to-play MMORPG there can be hundreds of different reasons for why a player decides to spend money on a virtual item or –currency. It can therefore be hard to hypothesize over which of these reasons that cause players convert from being non-paying users to paying users. Similarly, in a Real-Time Strategy game (RTS), we might wonder which order of constructing buildings for a base that is the most effective? In any type of game, it is interesting to know what types of behavior our players exhibit. These are all examples of explorative questions.

A typical data-driven method for explorative research is the **drill-down analysis**, where you examine the gameplay metrics data at more and more detailed levels until an answer is found (see Chap. 12 for details, or: Han et al. 2005).

Hypothesis-driven metrics research is when we are looking to confirming conclusions or ideas, or when we can predict the answer. For example, we may think that Spiders are way too powerful on level 12, and perform metrics analysis in order to confirm this suspicion, finding that either we are right or wrong in our hypothesis (wrong in the case of the only spider in the level being an arthritic tarantula with bad eyesight). Alternatively, we could have a hypothesis stating that the amounts of player deaths on a certain map correlates to the perceived difficulty level of the map. Checking metrics data on player death events with feedback from research study participants can either lead to confirmation or rejection of the hypothesis, possibly leading to the formulation of a new hypothesis. A commonly applied method in game data mining to answer these kinds of questions is prediction analysis – the application of specific algorithms to predict something, e.g. which users that will convert to paying users, or when a person will stop playing (Mahlman et al. 2010) (see Chap. 12 for an example).

In practice, as soon as you move outside of the kind of questions that can be answered with synthesis, a quick analysis or standard algorithms, e.g., “what is the number of active users today?” or “what is the average playtime for level 22?” (see also Chap. 4 for examples of common metrics based on synthesis), you end up mixing hypothesis-driven and explorative work, something that is also evident in the case examples presented here.

In our personal experience – and we fully acknowledge that others may have a different opinion – the explorative questions are usually more time-consuming to answer and more often requires analysis than the hypothesis-driven, specific questions, which can more often be handled using synthesis (or very simple statistical analysis) of the relevant gameplay metrics data.

Purely explorative questions are, again in our personal experience, not as common in game development. This both because there is usually some ambition with an investigation, and because we cannot be certain that explorative questions lead to a result that impacts on the quality of the game. This is not to say that purely explorative analysis of gameplay metrics data cannot be useful, but it is often a kind of blue-water research that it can be hard to justify expenditure on. This however presents an excellent opportunity for game companies to collaborate with academic institutions like universities, who are able to focus on research and can bring expertise to the table which companies may not be interested in or cannot afford, to contain on a permanent basis.

Another fundamental question in gameplay metrics work is whether the analysis in question is initiated and driven by **designers** (including leads and producers) or by **games user researchers**. As outlined in Chap. 3, often designers require feedback on specific questions related to game design or the user experience. In comparison, user researchers also conduct more explorative evaluations. There are other groups of stakeholders that can drive metrics analysis, but usually these are not as focused on gameplay. For example, marketing differentiates from designers and user researchers by focusing on players as customers, i.e. sources of revenue.

Finally, gameplay metrics analyses can operate with **small** or **large** amounts of data – these both in terms of the number of players involved in the analysis and the number of variables (or features) for each player. The goals of small-scale analyses are typically more detailed than large-scale work, where the sheer amount of data and players lends itself better towards drawing broad conclusions and evaluating overall patterns in player behavior.

The reason that these considerations about the fundamental ways we can approach gameplay metrics analysis in games are important and relevant for GUR is threefold:

1. It provides the means for classifying methods.
2. It provides a terminology to use when we talk about gameplay metrics work on a daily basis.
3. It provides guidance on which questions that are useful to answer when planning to address a particular problem or task – for example, considering whether a problem is best solved analytically or using simple synthesis, if a spatial approach will provide added depth or be irrelevant, etc.

In this chapter, we have included four cases that represent some of the fundamental differences in the approaches that can be selected when working with gameplay metrics. These are outlined in Table 14.1, which describes how each case example varies along the dimensions outlined in this section (complexity – synthesis or analysis, approach – explorative or hypothesis-driven, stakeholder initiating the investigation, scale of the dataset used, and timing in relation to launch of the game).

Table 14.1 Classification of the four case studies presented in this chapter. This way of classifying approaches towards gameplay metrics analysis is based on concepts that are foundational in science theory, e.g. synthesis and analysis. The above approaches are not exclusive to gameplay metrics; they could be applied to other research practices and methodologies as well. The column “Initiated by” refers to the stakeholder group in a game development company who initiates the metrics study in question. The stakeholders who benefit from the work can be another group than those initiating a study (e.g. marketing, management, design, user-research, producers)

Number	Case	Complexity	Approach	Initiated by	Scale	Timing
1	Weapon usage metrics in Kane & Lynch: Dog Days	Synthesis	Explorative	User research	Small	Pre-launch
2	Gameplay analysis in Fragile Alliance 2	Analysis	Hypothesis	Design	Large	Pre-launch
3	Frustration analysis in Kane & Lynch: Dog Days	Analysis	Explorative	User research	Small	Pre-launch
4	Causes of death in Tomb Raider: Underworld	Analysis	Explorative	Design	Large	Post-launch

On a final note, gameplay metrics work is often either **spatial** or **non-spatial**, i.e., we commonly work with data that are separated from the gaming environment, or with data that contain some sort of spatial reference. Chapter 17 discusses spatial game analytics in more detail. A similar point can be made about **static** vs. **dynamic** visualizations. Metrics reports – and GUR reports in general – can be represented either as static or dynamic reports, the latter being interactive so the stakeholder it is intended for can manipulate the report. Chapters 18 and 19 discuss visualization in more detail.

14.4 Case 1: Weapon Usage Metrics in Kane & Lynch: Dog Days

Kane & Lynch: Dog Days is a third-person shooter developed by IO Interactive in 2010. This case study shows how metrics was utilized during mid-development as a direct part of the development process and joined with several other sources of user-data. The case study represents an initially explorative approach, with synthesis of data from a small sample size and a situation where the user researcher initiates and runs the investigation.

In *Kane & Lynch: Dog Days*, as in other shooter games, controlling the hero’s weapon range is only one of many ways of controlling the player’s behavior – and experience. For instance, hypothetically, short-range weapons against far-away enemies will drive the player forward, while long-range weapons typically afford a more sniper-like and less roaming behavior. Since short-range weapons lead to

short-range kills, the killing experience is potentially more intense (closer to the frontline), while long-range weapons can give a more marksman-like experience, which can be equally satisfying. The trick is to, of course, enable the right experience at the right time. Balancing the weapon design in this regard can be hard to do based on observation or hypothesis alone, and players can have difficulties describing their ‘weapon behavior’ in abstract terms, since hopefully players will use the weapons as tools in a ready-to-hand fashion. Here, weapon usage gameplay metrics is useful to visualize allowing us to explore how different weapons can increase the likelihood of different behaviors.

During mid-development of *Kane & Lynch: Dog Days*, the user research team at IO Interactive wanted to explore how weapon attributes affected behavior on a specific prototype level. This intention was based to some degree on different vague hypotheses; however, the approach was to explore and learn. Playsessions were run with a small sample, only three participants, and various gameplay metrics captured via the Square Enix Europe Metrics Suite (Canossa and Drachen 2009), the system in place at Square Enix to log, transform and store gameplay metrics data.

The metrics data was plotted on a map of the prototype level. The level consists of two sectors (1 and 2). The map was produced using a native metrics visualization system developed in-house at IO Interactive, called *QuickMetrics*. The visualization system is hard-coded, and thus less advanced than some commercial-grade visualization systems and applications, e.g. most major Geographical Information Systems (GIS) packages such as *ArcGIS* and *QGIS* (Longley and Goodchild 2005; Drachen and Canossa 2011), or systems such as Square Enix’ other metrics visualization system *Amber* (developed by Square Enix Montreal) or visualization systems used by other publishers, e.g. EA Games, Microsoft Studios Research (Kim et al. 2008). However, *QuickMetrics* provided a very fast visualization process, enabling data (synthesis) immediately after a user-testing session.

At first glance, it looks like the behavioral pattern in the beginning of the map (Fig. 14.3, bottom left corner: *Sector 1*), is different from the behavioral pattern at the end of the map (Fig. 14.3, top-right corner: *Sector 2*). Some kills in Sector 1 (orange lines) are done at shorter distances and from more locations than the kills in Sector 2 (red lines). Other kills in Sector 1 (turquoise lines), on the other hand, looks like they were almost exclusively made from the same spot, though at a relatively short distance. In order to understand the behavior in the two sectors, we added additional information to the synthesis, namely movement paths; health level and player deaths (example shown in Figs. 14.4 and 14.5).

Comparing the beginnings of Sector 1 and Sector 2 (Figs. 14.4 and 14.5), we see that the player seems to be much more under pressure or at least vulnerable (the path color is more yellow and red) in Sector 1 than in Sector 2. This is also reflected in the number of player deaths in the two sectors. Exploring the synthesis of data represented by the spatial visualizations creates a backdrop for making a new hypothesis. This hints at a possible advantage in grouping data points permanently via the *QuickMetrics* tool, for instance by creating a new permanent metric called



Fig. 14.3 Enemy kills on a *Kane & Lynch: Dog Days* level. Green dots are the player positions at the time killing shots were made; the *colored lines* represent the lines of fire (different colors are different weapons), and *red dots* are locations of enemies at the time of death (Reprinted from Drachen & Canossa 2011; image is © Inderscience Enterprises Ltd.)



Fig. 14.4 Sector 1: (*left*) Enemy kills; (*middle*) location vs. health; (*right*) locations of death. The location vs. health map shows the hero moving around. If the line is *green*, the hit-point loss is from 0 to 80%, if the line is *yellow*, the hit-point loss is between 80 and 95%, and if the line is *red*, the hit-point loss is between 95% loss and death (Reprinted from Drachen and Canossa 2011; image is © Inderscience Enterprises Ltd.)

‘kills/location ratio’, i.e. how many kills are made per spot in a given area. If there are many kills per spot, this means that the behavior hypothetically is much like in the start of Sector 1 and 2. If we also add kill distance as a variable in this metric, then we can also highlight where on the levels we are more likely to have ‘under

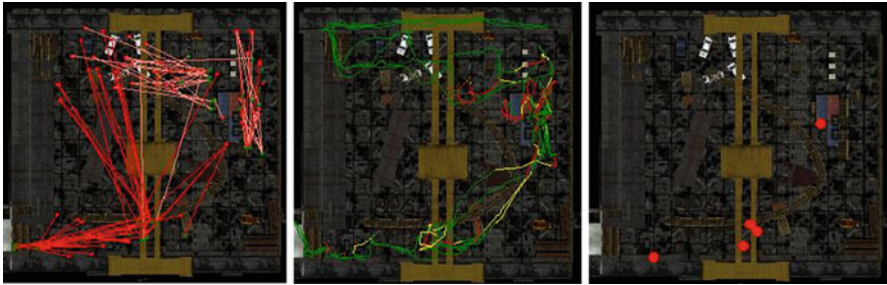


Fig. 14.5 Sector 2: (left) Enemy kills; (middle) location vs. health; (right) locations of death. The location vs. health map shows the hero moving around. If the line is *green*, the hit-point loss is from 0 to 80%, if the line is *yellow*, the hit-point loss is between 80 and 95%, and if the line is *red*, the hit-point loss is between 95% loss and death (Reprinted from Drachen and Canossa 2011; image is © Inderscience Enterprises Ltd.)

pressure'-experiences, rather than 'sniper/marksman'-experiences. This hypothesis could then be validated with a larger sample size. Obviously, the end-result of an analysis is not a nice looking heat map. The real goal is understanding the player's experience. Hence, any metrics analyses need to be validated with other methods as well to confirm the hypothesized link between behavior and experience. Likewise, AI movement (as a parameter) was not part of the synthesis, so this would also be beneficial to include in the work.

In this case, it was the designers' intention to put the player under extreme pressure in the beginning of Sector 1. In the ensuing analysis (which also drew on more qualitative data from the user test in question) it became apparent that the combination of the high-intensity, under pressure, frantic short-distance kills and high level of roaming in Sect. 14.1 and the less roaming, "pwnage killing" in the start of Sect. 14.2 created a compelling game experience coupling high stress and challenge with an opportunity for the player in Sector 2 to feel competent and de-stress by just staying in the same spot.

14.4.1 Mini-case: *The Perfect Path*

During development of *Kane & Lynch: Dog Days*, the lead level designer asked the games user research-team at IO Interactive to investigate whether players generally followed the path through the levels in the game as intended by the designers. This is an example of an explorative question in the spatial domain of gameplay metrics analysis, and a topic that can be tackled either at large or small scales.

The initial challenge was to figure out how to answer the question in the first place. Initially, the level designers were asked to play through the game in the way they intended, and their in-game behavior was logged via telemetry. This provided

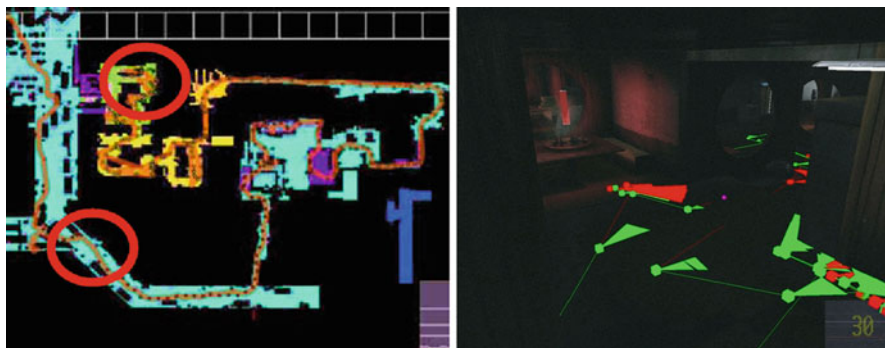


Fig. 14.6 (left) a section of a *Kane & Lynch: Dog Days*, level during development of the game. The brown line is a 2 m wide “perfect path”, i.e. the ideal or intended path through the level. Overlain the second-by-second location of a playtester, with red circles highlighting where the player deviates from the intended path (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)¹ (right) Metrics added directly into the game editor. Green color marks the player being in good health, red color heavily damaged. The cones indicate the direction the player is watching at the time. Individual points are 1 s of gameplay apart

a baseline to compare the behavior of players with what was intended in the design. However, obviously some deviation from the “perfect playthrough” of the designers was expected, notably in terms of weapon selection, strategies for overcoming enemies, etc. It was chosen to focus on the physical path described by players as they navigate the level. The path of the designers was overlain on level maps, and extended to be 2 m wide to allow for minor deviations (the brown path in Fig. 14.6, left). Subsequently, paths from user research sessions were overlain the path, and deviations from the designer-intended path located via overlay analysis (the paths were put on top of each other. Places, where the players’ path deviated within a 2-m wide measure from the designers’, were marked). The result pointed to specific areas where players strayed considerably from the ideal path.

This kind of explorative analysis showcases the potential of behavioral data to enable modeling of player navigation behaviors through the game environment. As a follow-up analysis, we used the same approach to locate areas where players were at low health, jumped, crouched, etc. (Fig. 14.6, right) This is of key interest to game designers because it allows them to observe from different perspectives how their games are being played. On a final note, navigation data can also be displayed directly inside the game editor, allowing designers to see through the

¹ACM COPYRIGHT NOTICE. Copyright © 2011 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481 or permissions@acm.org.

eyes of the player, with the added benefit of having recorded metrics mapped in the game environment.

14.5 Case 2: Gameplay Analysis in *Fragile Alliance 2*

One of the key design concerns when creating scenario-based FPS-levels is to ensure that the right events take place in the right locations. During the development of *Fragile Alliance 2* – a team-based multiplayer game mode from *Kane & Lynch 2: Dog Days* – the designers at IO Interactive wanted to know if players died in the right locations with the right frequency as intended by the design of the game. The hypothesis driving this piece of GUR work was that the design worked as intended in terms of the spatial and temporal behavior of the players. The investigation performed to address this question represents a hypothesis-driven analysis (either the players die in the right locations with the right frequencies or they do not). We had a large sample size to work with (thousands of death events from playtests). In this situation, the designers of the game initiated the investigation with an overall focus on map dynamics in a 3D team-based FPS context.

Fragile Alliance 2 (Fig. 14.7) pitches players as either mercenaries trying to accomplish a heist, or police trying to prevent this from happening. A game session will typically consist of multiple rounds being placed on the same map (scenario) and/or different maps. The winner of a round is the player who leaves with the most money, irrespective of how these rounds were obtained. *Fragile Alliance 2* features scenarios deeply integrated in the individual maps, and has a twist unusual in



Fig. 14.7 Screenshot from *Fragile Alliance 2*, showing a Traitor clearly marked to the remaining mercenary players (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)

team-based shooters, which is that mercenaries can kill each other and steal each other's loot. If, for example, a mercenary player managed to secure a sum of money from a bank vault, another mercenary could kill the first, and steal his/her money. If a mercenary kills another mercenary, he becomes a "traitor", and is clearly labeled so to the other players, but is allowed to keep all the money collected without sharing. If a mercenary dies, they respawn (are reinstated in the game universe) as police officers, working along with a group of AI-bots.

The game design is intended to create a situation where the balance of power initially will typically be on the side of the mercenaries, but shift towards the police (AI and players), putting increasing pressure on the mercenary team. After the second death, the player will typically not respawn, but will have to wait for the end of the game round (usually after a few hundred seconds depending on the map). Mercenaries run the risk of being killed by AI-police, police players and other mercenaries. If a player kills a traitor they receive an instant reward; however, if a police player kills the traitor that killed him as a mercenary, he will reap a bigger revenge-reward. If a police player secures an amount of loot money from a mercenary, he keeps a percentage as a finder's fee. The purpose of the mercenaries is to escape with as much money as possible – either by working independently or together.

In order to address the question posed by the designers at IO Interactive (did players die in the right locations with the right frequency, as intended by the design of the game), we ran a series of analyses on metrics data obtained from comprehensive user-testing. Focusing on death events (where players die on the *Fragile Alliance 2* maps and why) provided a means for investigating spatial behavior of the players, which is easier to work with computationally than path analysis. When, where and why a player dies holds a substantial amount of information about the spatial dynamics of a team-based shooter level, as is known from traditional applications of heat maps, and this consideration formed the basis for the investigation.

One of the maps (or scenarios) in *Fragile Alliance 2* is the "Subway". The map gives the mercenary team the job of reaching and breaking into a subterranean vault, and then escape through a street-level area. The police team is tasked with preventing the heist. For the "Subway" map to work as intended, we would expect the mercenary team (or survivors) to at least in some cases reach the exit and complete the mission, rather than being gunned down by the police early in the map.

The dataset used for this analysis contained roughly 38,000 death events, obtained from sessions where the designers played the map "for fun", without any specific testing purpose in mind. They thus represented the closest we could get to "natural" player behavior data at the time. Locations of the death events were extracted along with various other variables deemed appropriate (e.g., role of the player at the time of death, role of the killer).

A central advantage of the *QuickMetrics* tool that was mentioned earlier is that the visualizations are pre-defined, and data from user-sessions can, therefore, be rapidly displayed and used in context with other data sources – or even discussed with the player. The system (in its current version), however, does not provide summative in-depth data analysis. It can provide visualizations of spatial data, but it does not provide opportunities for more complicated processes. In order to

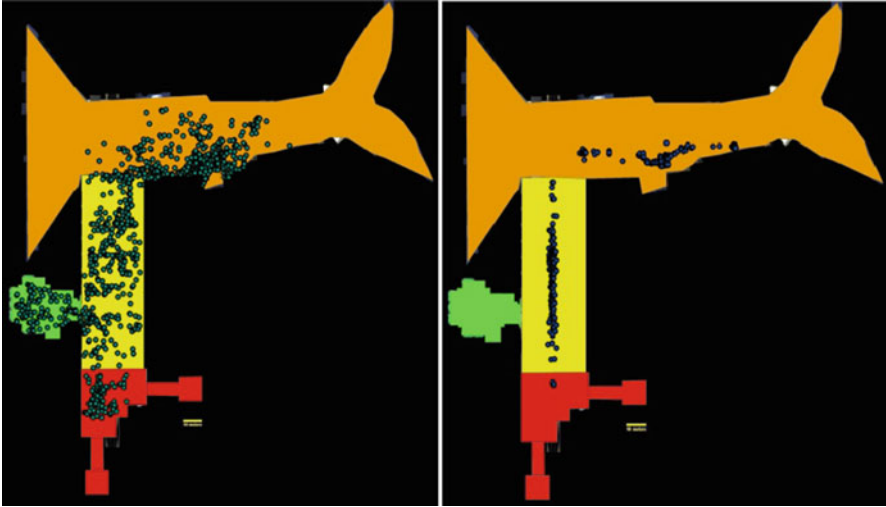


Fig. 14.8 The “Subway” map divided into four sub-sections: *Red*=spawning area for mercenary players; *Yellow*=subway; *Green*=vault area; *Orange*=road/exit area (+ spawning area for police players and police AI). (*left*): The locations where police officers were the cause of death. A broad distribution is apparent indicating that AI police officers can reach the entire map. (*right*): Locations of environment-caused death events. The events in the *yellow* sector of the map are caused by players being run over by a subway train while crossing a set of tracks, while the death events in the orange zone are caused by exploding scenery (e.g. cars that explode after becoming too damaged by weapons fire) (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)

perform custom spatial analyses, we worked with geodata (gameplay metrics with a spatial component – point, line or polygon), and used a Geographical Information System (GIS) as the front-end tool. A GIS is inherently flexible, but also time-consuming to use (see Chap. 14 for more on GIS software). Thus, it is ideal to have any analysis/visualization that can be pre-defined moved into a custom application, in our case *QuickMetrics*, for ease of use and to increase the speed with which visualizations can be generated (see Chap. 7 for another example of a visualization tool).

We used *ArcGIS*, an off-the shelf package. In *ArcGIS*, spatial data are added as individual layers (see Chap. 17 for more on the process of generating data layers in spatial analytics). In the current case, one layer was developed for the zonal division of the “Subway” map and for the locations of death events as a function of player role at the time of death (e.g. mercenary, police, traitor etc.). The “Subway” map was divided into four sections according to the level design (show in Fig. 14.8). In the map the mercenaries spawn in the bottom of the map, the police AI agents to the top right (Fig. 14.8). The objective of the mercenaries is a vault, located to the left in the map, and thereafter to reach the level exist, in the top right corner, behind the spawning area of the police. The game level consists of four sub-sectors:

- The **spawning area**, where the mercenary players enter the game and begin addressing the heist objective (red in Fig. 14.8).
- The **vault area**, where the money they need to steal are located (green in Fig. 14.6), a **subway station** area approximately in the middle between the spawning area of the police (AI and players) and mercenaries (yellow in Fig. 14.8).
- The **exit area**, an area at street level (orange in Fig. 14.8), through the rightmost side of which the mercenary players go if they want to escape the map (i.e. complete the heist or run away).

Dividing the level map into sub-sectors permits a more detailed analysis of the distribution of the death events. By combining visualizations of the spatial behavior of players with statistics of their temporal (and spatial) behavior, a more thorough understanding of the player behavior is gained. The analysis we ran showed that mercenaries primarily turn traitor in the beginning of the game in the spawn area, but most commonly (55.72%) in the road/exit area – i.e. when the mercenaries are close to getting out with their stolen booty. For the mercenaries, the majority of the kills occur in the spawning area, where mercenaries enter the game (red zone in Fig. 14.8). The AI police kill and death events are spread across the entire map, indicating that their search & destroy behavior is working excellently (Fig. 14.8 shows the locations where AI police kill opponents – distributed across the map). Suicides, i.e. when a player dies for a reason not related to another player, occur in the vast majority of cases (76.04%) in the road/exit area, where a series of cars are placed which can explode, doing damage. A smaller part takes place in the metro station area, where players can be hit by metro trains while crossing the tracks coming from the vault to the exit/road area to the north in the map (shown in Fig. 14.8 (right) as a line of death events up the middle of the yellow area).

That does it for the role of the killers, but what about the role of the players when they die? Police (recall that these can include both players and AI) are often killed in the road/exit area where they spawn (69.32%) (shown with purple bars in Fig. 14.9), and very few are killed in the spawn and vault areas, where instead the mercenaries are under pressure from the police (44.17%, shown in Fig. 14.10). Interestingly, traitors are typically killed in the spawn area (61.25%) (shown in Fig. 14.10), but rarely in the road/exit area (8.81%) (shown in Fig. 14.10), which in combination with spatiotemporal analysis indicates that it is a much more risk-filled endeavor to turn traitor early in the game rather than later (associated spatiotemporal analysis shows that mercenaries turning traitor outside of the spawning area rarely move into the spawning area again – by this point the action has moved to the other segments of the map).

We also considered these patterns in a temporal context. In *Fragile Alliance 2*, one of the key design components is that there should be a shift in balance as the game progresses. Initially, the mercenaries are strong, however, as more and more mercenaries are eliminated, the police force – augmented by AI agents – became stronger. In order to evaluate if this shift in power actually manifests, a time-series analysis was performed using temporal slices, finding to the pleasure of the

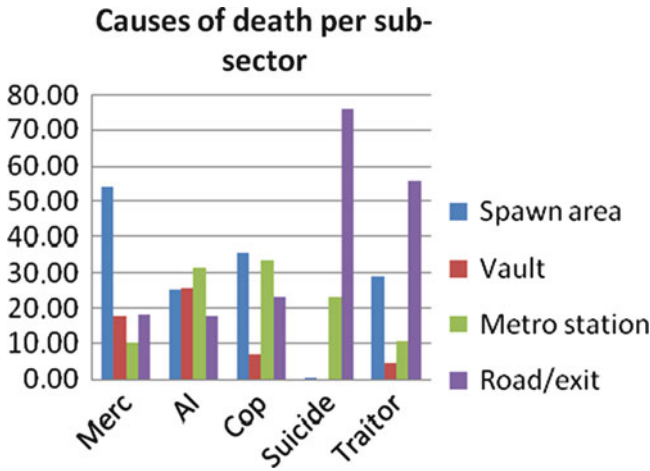


Fig. 14.9 The causes of player death as a function of the four sub-sectors (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)

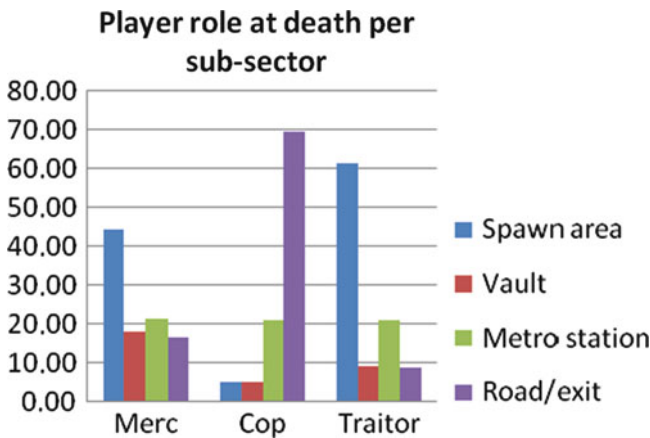


Fig. 14.10 The role of the player at the time of death as a function of the map sub-sector (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)

designers that there is a significant shift in who the dominant killers are from the early to the middle of the typical game round. This is illustrated in Fig. 14.11, where the left pie chart shows the breakdown of the causes of death (killer roles) summed for the first 45 s of play, and the right pie chart the causes of death, but following 90 s of play until the end of the game round. A distinct shift in the causes of death is apparent.

While the behavioral patterns of the players indicate the manifestation of the designer’s intent, a somewhat larger amount of death events occur in the spawn area (lowermost part of the map) than is ideal, which could indicate that

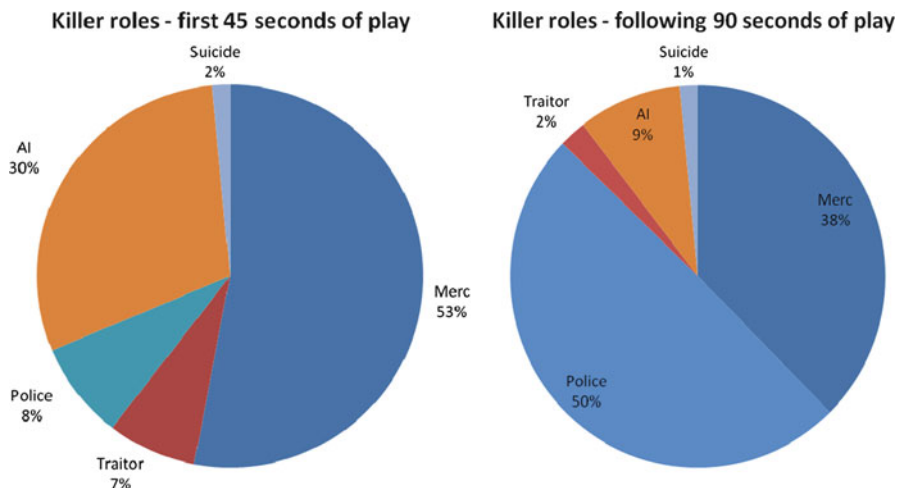


Fig. 14.11 Roles of players who kill another player or AI-bot during the first 45 s vs. following 90 s of play on the “Subway” map. A distinct shift is noted as mercenaries are eliminated and the police force gains correspondingly in strength

mercenaries are perhaps a bit too eager to turn traitor early in the game. This is exemplified in Fig. 14.12, where death event data from just one sessions are shown. Out of 253 kills, 119 appear in the spawn area for the mercenary team (blue area in Fig. 14.12). This kind of behavioral pattern is however not necessarily a problem to the user experience – it can be the opposite: the pattern observed may be fun to the players even though it is not what was expected. To find out requires the use of other GUR methods than telemetry analysis, for example play-testing using a think-aloud protocol or surveys (Isbister and Schaffer 2008; Lewis-Ewans 2012). For an introduction to user research methods in general, including think-aloud testing, see Kuniavsky (2003).

14.6 Case 3: Frustration Analysis in Kane & Lynch: Dog Days

The third case is an example of a small-scale, explorative analysis where the user research team at IO initiated and drove the investigation based on a problem observed during user testing: During the regular user-oriented testing of *Kane & Lynch: Dog Days*, in the later development stages of the game where vertical slices were available and the game is mostly functional, the user research team at IO Interactive observed several participants becoming obviously frustrated while playing a specific segment of the game (e.g. groaning, throwing the controller away, agitated, angry etc.). This case study describes the work conducted subsequently to identify which

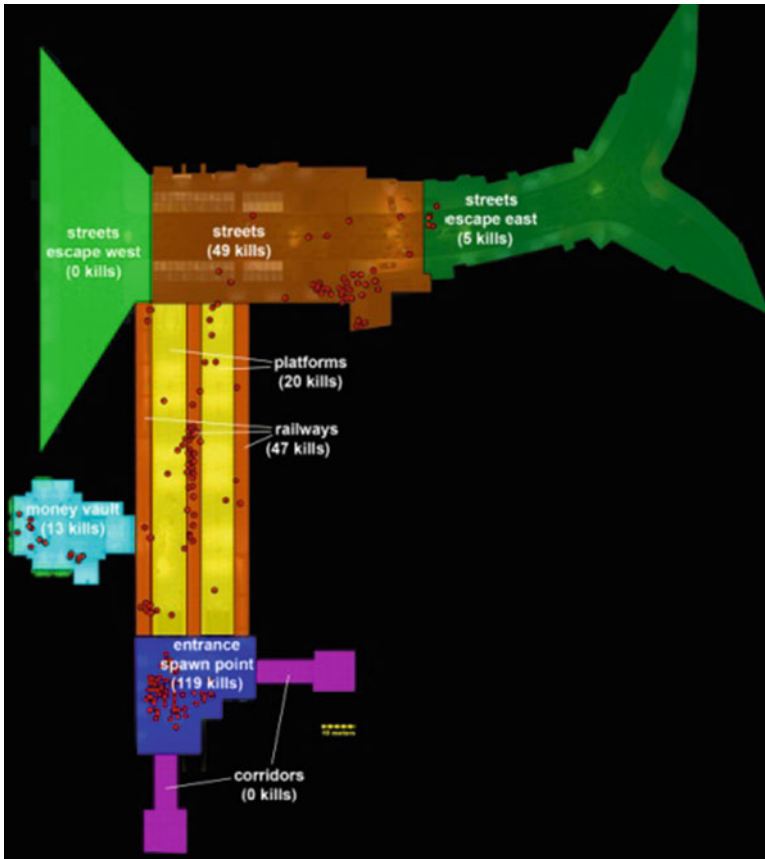


Fig. 14.12 Visualization of death events from the “Subway” map. The map shows the distribution of 253 player death occurrences (all from a specific play session) overlain the level map, and has added explanations to guide the interpretation of the map (Reprinted from Drachen and Canossa 2009a with permission; image is © 2009 by ACM, Inc.)

behaviors led to the exhibited frustration. The case study is described in more depth in Canossa et al. (2011).

A central limitation of gameplay metrics analysis is that the data can inform what players are doing, but cannot directly answer questions about ‘why’ players are exhibiting that behavior, although sometimes we can make decent guesses (inference). The same is the case for the user experience, we cannot directly tell what the player feels like. This is precisely the reason why the more design-oriented use of gameplay metrics-based analysis cannot always stand alone in user research, and is ideally combined with other approaches for GUR, e.g., observation, interviews or questionnaires as discussed in Chaps. 21 and 22. The following case study presents another example of how game analytics and traditional user testing can go hand-in-hand to inform game development.

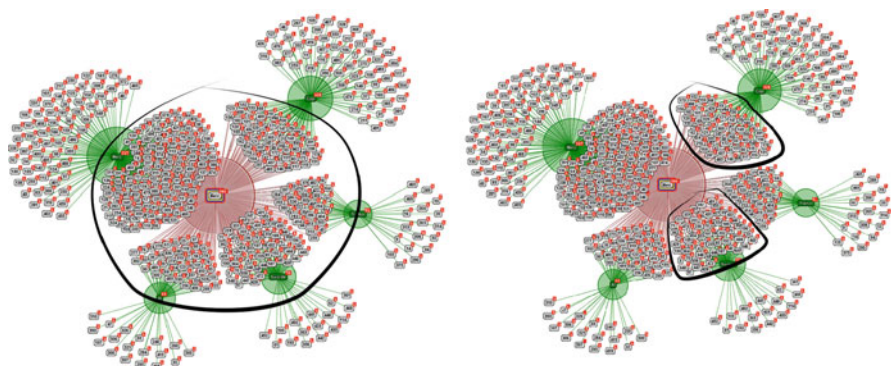


Fig. 14.13 “The Flower of Death”. This is an example of a more interactive way of working with gameplay data that was tested back in 2008 at IO Interactive. TouchGraph Navigator is a simple visualization tool which allows the user to visualize connections between data points. In this case, 253 death events are shown, including the role of the killer and victim. TouchGraph Navigator permits the manipulation of the groups using drag and drop, which makes the data easily navigable. In this case, the tool was used to discover that a significant percentage of the mercenaries (the *central mass, circled* in the diagram to the *left*), were killed by suicides (i.e. environmental effects) (the *lower right mass* of data points marked on the *right diagram*). Checking the spatial distribution of these events, it was found that most of these kills occurred because the players were hit by a train in the railways area of Fig. 14.12. A signal was added to the train, warning of its arrival, which solved the problem. The *upper right* group marked in the right diagram above are mercenaries killed by police AI or players. The amounts of deaths is about the same as the suicides, suggesting that this early stage of development, the danger to the mercenary players needed adjustment

In many user-testing situations, the causes of player frustration are related to bugs, unfinished designs – malfunctioning items, checkpoints etc. Thus, finding the reason for player frustration can be a straight-forward process of observations and interviewing, ideally assisted with videos of screen capture and similar tools. However, what the user researcher team observed in this case was different, they identified a trace of specific patterns of behavior connected to frustration, which they believed was connected to a ‘real’ design issue. For example, they observed players rushing forward within the game after dying over and over, not really looking around for danger, increasing the risk of dying again, causing even more frustration, after which the participants slowly regained their composure and concentration and took a more careful approach.

When running user tests with one participant, detecting such behaviors and querying the player about them is possible. However, in the context of larger-scale tests (e.g. 5 or 25 players), it becomes cumbersome for resident GUR personnel to detect the various hints in the behavior of players that point towards frustration. This prompted the idea of investigating if particular expressions of frustration were related to particular behaviors while playing the game (avatar/character behaviors) and subsequently using gameplay metrics data to pinpoint the exact locations in the game world where such behaviors occur, across multiple players.

The first step was to examine the detailed telemetry data for one of the frustrated players (Fig. 14.14) in conjunction with the video recordings of the screen and the



Fig. 14.14 Visualization of the spatial behavior of a player through the environment of a *Kane & Lynch: Dog Days* level, between two death events. The location of the player is plotted at each second of playtime, and a color scale applied to show the dimension of time along the path. Various events are plotted as symbols: enemy kills (*blue dots*), weapon pickups (*red triangle*) and taking cover (*green squares*). Spatial metrics visualizations such as this one are highly useful for the detailed evaluation of gameplay and balancing in shooter games (Reprinted from Canossa et al. 2011 with permission; image is © 2011 by ACM, Inc.)

player from the testing session. This was done in order to examine what the player actually did during the periods where he exhibited frustration, identified through body movements, facial expression and verbalizations. The metrics data were fairly comprehensive. We used a temporal sampling interval of 1 s. The data includes the location of the players, the vector of the avatar, the vector of the virtual camera, the health of the player, movement modifiers (walking, running, sprinting), whether the players were crouching or not and whether the players were “in cover” or not. In addition, triggered events (events the logging of which are triggered by specific player actions) were analyzed, which include checkpoint activation, picking up weapons and ammo, making use of exploding objects in the level, being “down but not dead”, killing of Non-Player Characters (NPCs) and player deaths. Examining the data generated a list of indicators of frustration (Fig. 14.14), as follows:

1. The player dies repeatedly in the same location or even regresses in terms of progress made between each death event.
2. The number of enemies killed decreases considerably with each successive attempt to progress in the game following a death event.
3. The pace of movement becomes considerably faster for each attempt, and the same route is taken each time.

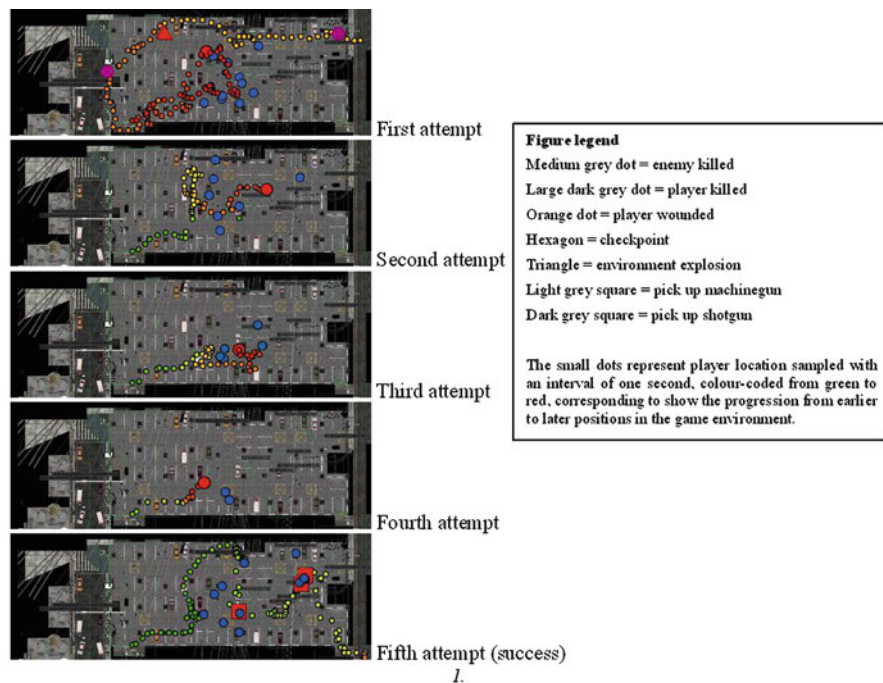


Fig. 14.15 The images show the path of a single player (participant) and specific events that occurred during the gameplay session used in the case study. Each image represents the time segment from one instance of player death to the next, showing decreasingly less progress in the game from death 1 to 4; indicative of a behavioral pattern pointing towards player frustration (*red dots*: player deaths, *blue dots*: NPC kills, *red square*: weapon or ammo pickup, *red triangle*: environment explosion, *purple dots*: checkpoints, *small dots*: player position in time) (Reprinted from Canossa et al. 2011 with permission; image is © 2011 by ACM, Inc.)

4. There is minimal or no use of special abilities, picking up of weapons or using the environment for help, e.g. triggering explosions.
5. The vector of the camera increasingly coincides with the direction of movement of the character – the higher the frustration the less interest in examining the environment.

A second and a third set of metrics data were also examined, based on reports from the participant about experienced frustration. The behavioral patterns in these situations matched those located initially (Fig. 14.15).

Following the identification of the above list of possible indicators of frustration, we analyzed metrics data from a sample of 22 randomly selected players among the testers used by IO Interactive. We used procedural algorithms to confirm the initial results and establish whether the pattern of behavior identified signified frustration within the 22 selected play sessions. It should be mentioned that it is, of course, possible that other players might also have experienced frustration but reacted very differently.

We found a match in the behavioral patterns of 6 out of the 22 players. One or more times during the playtest, these six players exhibited the same kind of behavioral profile as defined by the five points listed above. This profile is radically different from the kind of behaviors observed in the remaining 16 players.

The behavioral variables were found to correlate, e.g., if a player death event (Pd) occurs within 2 min, the average pace (speed) of movement of the player (Pm) will increase compared to the average movement pace for the entire game, and the number of NPCs killed decrease progressively between each death happening. Similarly, the number of weapons and ammunition supplies picked up (WApu) decreased progressively as players continuously died within shorter time intervals.

Based on the behaviors of the six players and the initial tester, we developed a model specifying the timing and frequency of the behaviors identified, specifying the value of the key parameters indicating player frustration. The model was presented as follows:

```

tn <tf<tn+1
Pd>=2
0<Pd1<20
Pmf>Pm
NPCd (tfn)>NPCd (tfn+1)
WApu (tfn)>WApu (tfn+1)

```

Where:

- Timestamp (t). The timestamp is set to zero the moment a new play session begins. <tf> describes a time interval that has been identified as “frustrated”
- Number of player’s deaths (Pd), <Pd1> expresses location of player deaths in world units.
- Player’s pace of movement (Pm) measured as distance in space travelled in 1 s, averaged for the whole playsession. <Pmf> defines the average pace of player movement during an interval of time identified as “frustrated”
- Number of NPCs killed (NPCd).
- Number of weapons or ammo picked up (WApu)

Importantly, all conditions need to occur simultaneously for the model to contain all the indications of player frustration reported – i.e., frustration is not indicated by any single behavioral variable, but the occurrence of a set of behaviors (e.g. fast movement and many rapid death events).

As the final step in the investigation, the six players who exhibited frustration were called in for open interview sessions. Using video recordings of their play sessions and visualizations of their metrics data, we attempted to uncover if the players felt frustrated during the intervals identified by the model. During the interviews, we used a custom browser-based tool (Fig. 14.16), “G-player” to show animated replays of test sessions, which is very useful when reconstructing play experiences with players. The participants confirmed that in all of the segments of play identified by the model, they had experienced undesirable frustration (i.e. frustration contrary to the user experience).

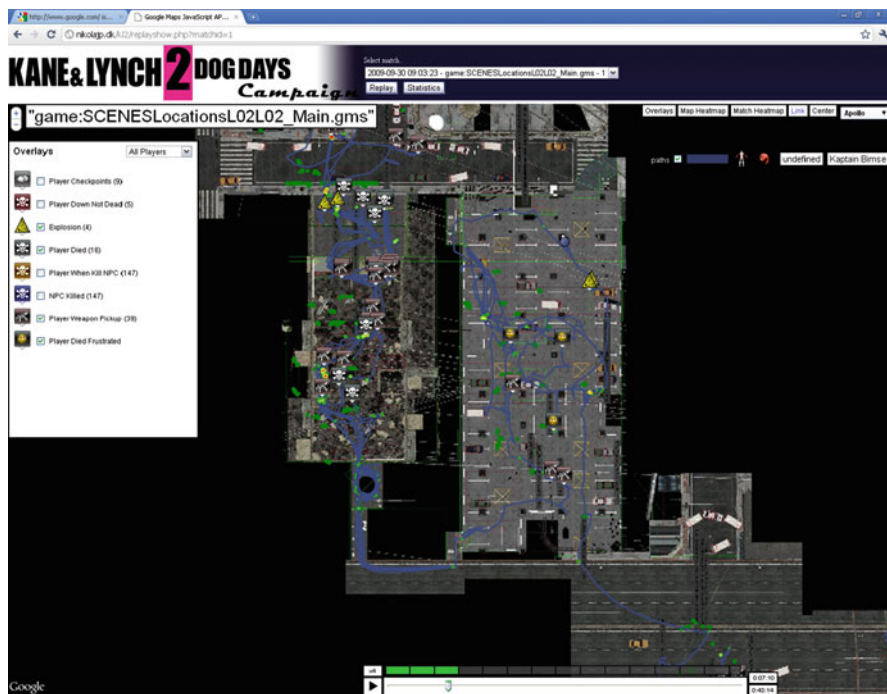


Fig. 14.16 The G-Player dynamic visualization tool, which allows replaying game sessions from a *top-down* perspective, showing the behaviors of different players and events as icons as they move around the playfield. A real-time version, where metrics data are being fed directly into G-player during the research study, is currently being developed (Reprinted from Canossa et al. 2011 with permission; image is © 2011 by ACM, Inc.)

The case study is an example of a relatively comprehensive metrics-based examination, more time-consuming than the day-to-day use of metrics data to evaluate game designs, but nonetheless potentially highly useful. The initial exploration led to a hypothesis that frustration is quantifiable and visible in the metrics. This led to the analysis of data and partial validation of the hypothesis.

There is much further work to be done to validate the utility of the model and its generality across different games. However, the case study does showcase the potential fruitful systemic interplay between hypothesis and exploration which potentially can make it possible for the user research team to more automatically detect and evaluate frustration in user research sessions, by analyzing the behavioral metrics data from the testers. This saves valuable manpower and provides a means for the development of automated frustration detection systems. Considering how fast and inexpensively automated the problem detection system operates, it provides a concrete benefit, because in a real-life user testing situation, it is not realistic to expect user researchers to keep track of all of these variables while running user tests. This highlights the usefulness of automated

gameplay metrics tracking and -recording as a tool for user-oriented testing. At the same time, it also highlights the challenge of automating. In other words, just because it works on this game and with these users, does not necessarily mean it is universal.

14.7 Case 4: Causes of Death in *Tomb Raider: Underworld*

This case study, originally conducted following the launch of *Tomb Raider: Underworld*, represents a designer-driven, large-scale and fundamentally explorative example of gameplay metrics analysis. The focus of the case study is challenge. This is historically one of the key objectives that a game user research team investigates; the game should provide the exact right amount of challenge to the target audience. One way to get an initial grasp of this key question in a game like *Tomb Raider: Underworld* is to consider the locations and causes of player death. In essence, investigating areas where players die consistently and repeatedly may signify imbalance in terms of the challenge posed by the areas. Identifying such areas via gameplay metrics analysis provide valuable information about potential problem areas, directing user research on challenge.

This kind of design problem can be addressed during production, as well as post-launch. It also can be studied as a non-spatial and/or a spatial angle. In this example, the metrics data were collected following the launch of *Tomb Raider: Underworld*, which allowed the tracking of the entire population of players. Post-launch data analysis is excellent as it informs us about patches as well as provides information for future game productions. In an MMOG (massively multiplayer online game) context, post-launch analysis is essential to the continued development of the game (as is also discussed in Chaps. 4 and 7; and by Mellon 2009).

In *Tomb Raider: Underworld*, each game level is comprised of multiple “map units” for the purpose of metrics logging, about 100 in total. The Valaskjalf map unit is one of the more complex puzzle/trap locations in the game, featuring multiple different challenges to the players’ skills. In analyzing the patterns of death in map, the first step was to produce a heat map based on locations of player death (Fig. 14.17).

Heat maps can be produced in different ways, e.g., using density functions or simply summing the number of deaths occurring within grid cells (as is done here). Such heat maps are excellent for informing the lethality of different game areas. However, it is unspecific as to the nature of the deaths. In order to evaluate where different causes of death, such as death by falling, death caused by different kinds of environmental dangers, or death caused by AI enemies (and if they occurred as intended by the game’s design), a series of visualizations showing the areas where players died of different causes (Fig. 14.18).

We wanted to know where players died from a large number of different threats (and died a lot). Such areas potentially represent sites of high challenge for the

Fig. 14.17 Grid-based heat map of the locations of player death in the Valaskjalf map unit of *Tomb Raider: Underworld*. Scale ranges from *light green* (low numbers of death) to *red* (high numbers of death). Locations with no color have zero deaths. Four of the most lethal areas are marked with *red circles* (Reprinted from Drachen and Canossa et al. 2009b with permission; image is © 2011 by ACM, Inc.)

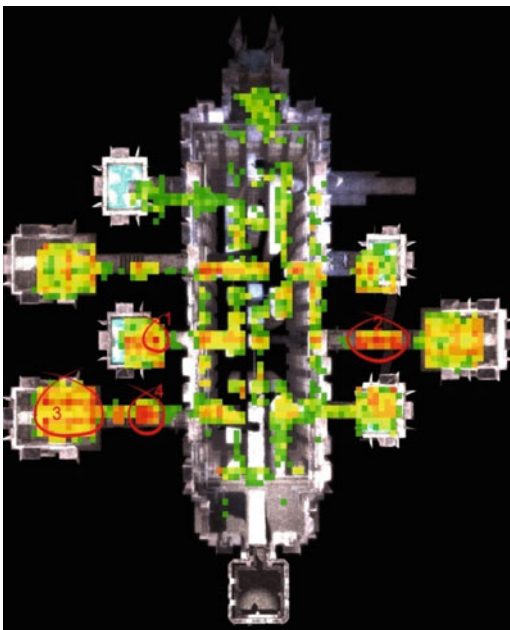


Fig. 14.18 The Valaskjalf map has been overlain with three layers showing the extent of three separate causes of death: Falling (*light blue*), traps (*green*) and water volume [players drowning by being submerged in rising water] (*red*) (Reprinted from Drachen and Canossa et al. 2009b with permission; image is © 2011 by ACM, Inc.)

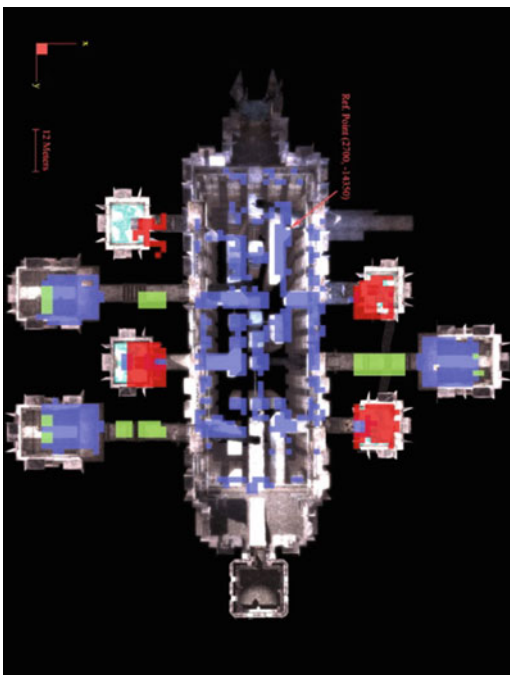


Fig. 14.19 Overlay analysis showing the areas of the map where the highest number of different causes of death occur, on a scale from *light green* (1–2) to *red* (6). The area with the most variety in causes of death is also one of the places with the highest overall death count (Fig. 14.18) (Reprinted from Drachen and Canossa et al. 2009b with permission; image is © 2011 by ACM, Inc.)



players, and therefore form targets for evaluation about whether their challenge level is too high. To some degree, these areas can be predicted from the game design. However, if there is one thing user research testing has shown game developers, it is that players are very hard to predict. So, in this case, we added a series of layers on top of the *Valaskjalf* map, each layer containing the spatial distribution of one cause of death. Using GIS, we performed a simple count across the layers (8 in total) (Fig. 14.19).

The result of the overlay analysis shows the distribution of death causes within the map. Four areas, however, showed several different causes of death (see Fig. 14.19). For Area 1, a high number of deaths occurs in one specific grid cell (about 5*5 m) caused by a low variety of causes: the attack of a Thrall (an AI-enemy, third row in Fig. 14.20) combined with a tricky jump (death by falling, fourth row in Fig. 14.20). If the number of deaths occurring in this area is deemed to be high (i.e. prevents or diminishes player enjoyment), the analysis suggests two ways of solving the problem: making the jump easier or eliminating the Thrall enemy.

Area 2 (second column, Fig. 14.20) also shows a high number of deaths and even though there are only two different causes: tarantulas (third row) and traps (fourth row), the distribution of tarantula kills on the *Valaskjalf* map is not spread enough to justify all the deaths displayed. This means that most of the deaths are caused by the traps. This could suggest that the traps should be more lenient. Area 3 displays a high number of deaths, which is motivated by a varied array of

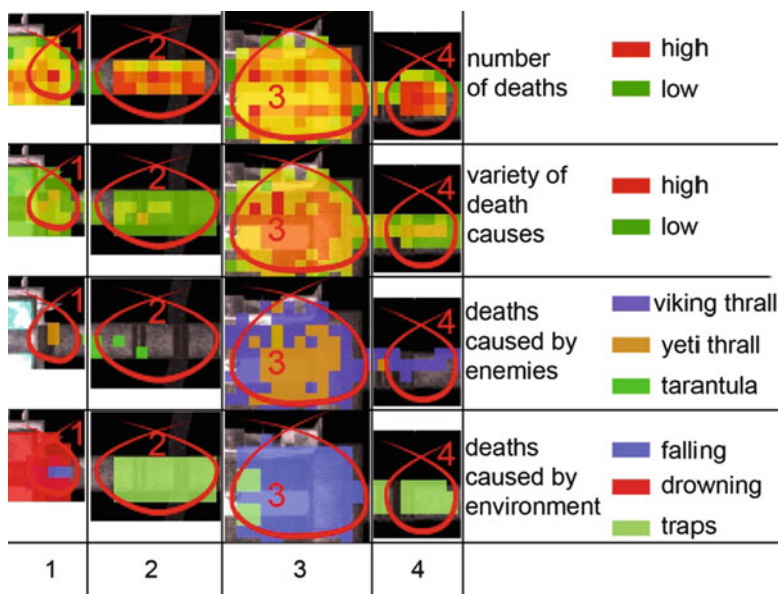


Fig. 14.20 Detail of the overlay analysis with a breakdown of the four targeted areas with multiple causes of death. Four *ArcGIS*-derived layers included: Aggregated death count, aggregated causes of death (*top two rows*) and deaths specifically caused by enemies or environment effects (*bottom two rows*) (Reprinted from Drachen and Canossa et al. 2009b with permission; image is © 2011 by ACM, Inc.)

causes: enemies, environment effects and falling – this is the climax of the level and clearly the toughest part to get through without dying. As with Area 1, a revision of the challenge level might be useful here – further attention is required. Area 4 displays very similar characteristics to Area 2 with similar implications in terms of the play experience.

The (admittedly rather simple) spatial analysis has thus identified potential trouble spots in the *Valaskjalf* map design, which subsequently can be analyzed in further detail. For instance, it is possible to compare the data with user-satisfaction feedback from the level to evaluate whether there is a problem or not. GIS permits different layers to be turned off and on flexibly, and also that specific layers be given different weights in the analysis – if, for example, players dying of electrocution is an unwanted occurrence in the game design, this can be given a greater weight and thus show up stronger in the analysis. Additionally, maps can be exported using an extension as dynamic reports, which permits the user to add or remove layers dynamically, forming the perfect reporting tool for giving feedback to e.g. designers. This type of analysis, even though explorative and time consuming to begin with, has the potential for being operationalized into automated metrics queries in the user research of new titles.

14.8 Working with Gameplay Metrics

There are a few experiences that we would like to pass on from our work with gameplay metrics and other approaches to GUR. Perhaps the key experience we have made is that a healthy user research system has been the continual interplay or feedback between different methods for investigating and researching games – and this includes game analytics.

Managing user research, to us, is very much concerned with keeping the movement and information exchange flowing between the different parts of the user research process, and allowing the user research system to learn. The same goes for working with gameplay metrics – it provides the biggest ROI when there is a continual openness to and constant interplay between work based on synthesis and analysis, exploration and hypothesis, user research and design, and small and large scale data sets, etc. For example, evaluating data from a single user research participant can provide important new information (as is known from usability testing where a small number of people can typically find the majority of interface errors), but will often need to be validated with data from a larger group. Similarly, the research-driven needs should be aligned with the designer (and business) goals, and conversely, designer-driven questioning should be aligned with GUR goals and methods. Additionally, method interplay is important, i.e., how the interplay between different approaches should be handled. This is not a mundane challenge since the theoretical foundations of different methods often clash significantly, as is for instance the case for usability (positivism) and participatory design (interpretive social science) (Silverman 1993).

Apart from these overall considerations, there are three other issues we would like to highlight:

- **Remember that gameplay metrics inform what players are doing, not always why.** Gameplay metrics provide information only regarding actions undertaken in-game by players, it is not directly possible to assess reasons and motivations behind the action, unless additional user data are captured – although inferences can be drawn. Gameplay metrics do not inform whether the player is having a good day, or what the player thinks of the game experience. In short, gameplay metrics cannot provide any contextual data. A metrics tracking tool can only record information from the specific game software. When an analysis of a set of metrics data point to a specific player behavior, it is therefore almost always a good idea to combine the approach with observations or user feedback (even if just some simple questions like how fun an encounter was, what they found to be the most frustrating/challenging, etc.). The same point is emphasized by our GUR colleagues in most of the conference presentations and talks we are aware of (see e.g. Chaps. 21 and 22; or Isbister and Schaffer 2008). This is part of the feedback loop that keeps the metrics tools relevant and useful. In addition to utilizing gameplay metrics in user-oriented testing, *IO Interactive*, *Crystal Dynamics* and other *Square Enix Europe* developers potentially involves a battery of methods, including audiovisual recording and analysis, survey-based approaches, expert testing, different forms of usability testing, etc., depending on

the specific requirements of the user test. The combination of gameplay metrics analysis with existing methods for user-oriented game testing provides the ability to probe player behavior and its causes in detail.

- **Find the right metrics to track at the start of the process.** Our recommendation is to involve the consideration of which gameplay metrics to track as early as possible. The earlier that the design and user research teams sit together and figure out what information to track, the better, with the understanding that you will always learn something new about this on the way. Also, you should consider whether data are to be logged post-launch, and if/how metrics should tie in with community feedback. There are no general rules about what should be logged; it all depends on the specifics of the game design. It is, however, vital to ensure that your logging system is flexible to accommodate the adding of new variables as the development progresses so as to make it more likely that exploration and hypothesis work is actually possible with the available data.
- **Manage the allure of numbers.** Gameplay metrics present hard numbers about player behavior, convincing diagrams and what not. However, critical thinking should always be applied – sometimes the analysis will show one thing through a flashing red color on a heat map or another suspicious pattern in the data, but the problem may actually rest in some minor detail in the design. The expertise of the designers is important to spot these kinds of problems, and this is another argument for why user-research and designers should work closely together. Also, and even more importantly: just because it looks good, does not mean it is true. Heat maps and graphs look cool and travel better in organizations than two pages of text with detailed explanation of a specific finding from a comprehensive user test. Heat maps, data visualizations and diagrams are deceptively easy to understand, but, also, they make it easy to ignore other factors that could potentially hold an impact on whatever is being investigated, but which is not included in the metrics-based analysis in question. Heatmaps can be printed out, provide valuable feedback on design, and also used as trophies on the wall of an office, or they can be powerful tools in the politics behind game development. So what is wrong with that? Nothing necessarily, except it can potentially take a life on its own and be used out of context, thereby escaping the feedback loop that is supposed to keep it in check.

14.9 Final Considerations

In this chapter we have attempted to provide some insight into a few simple ways to work with gameplay metrics in practice during mid-late production, in the specific context of single/multi-player, third-person 3D-games like *Kane & Lynch: Dog Days*, *Tomb Raider: Underworld* and *Fragile Alliance 2*. There are many ways to work with and utilize this highly useful source of user behavior data both during production and post-launch, and ours is just one of these. Apart from the differences between single and multiplayer games, the degree of non-linearity and whether the

game in question supports a persistent world or not, game type is also highly important. In essence, working with traditional boxed single/multi-player games is somewhat different from persistent-world massively multi-player online games and social online games (see also Chaps. 4 and 5), where a substantial focus is on the application of gameplay metrics analysis and synthesis to tune game design on a running basis, as well as for monitoring churn rates, average revenue per user and similar business metrics. Even within the confines of the shooter genre, there are different approaches to game analytics, but there appears to be a general consensus that gameplay metrics mesh well with other user-oriented approaches for evaluating and testing games.

Our experience has shown that more exploratively-oriented enquiries initiated during production – case 2 about finding out if the kill balance worked – being a good example, have a tendency to become part of the daily practices, and topics/issues that were regularly consulted by games user researchers and designers alike. While explorative work is often more time consuming and less certain to produce actionable results on than work driven by specific questions or hypotheses, the potential benefit can be substantial. It is in the interplay between different approaches where the best procedures and methods are developed. We emphasize an adaptive and flexible approach, having developed in-house tools and appropriated off-the-shelf software for the purpose, because we acknowledge that games are different. For example, there is a big difference in the metrics that are of interest in an RTS as compared to a soccer management game. The user research and analyses that we conduct during the production of different games varies.

In summary, gameplay metrics analysis addresses one of the major challenges to GUR, namely that of tracking and analyzing user behavior when interacting with the very complex systems that make contemporary computer games. As a user-oriented approach, it complements existing methods utilized in the industry very well, providing detailed and quantitative data to supplement qualitative and semi-quantitative data from other user research methods.

14.9.1 Next Steps

If interested in reading more about practical work with game telemetry in a game development context, or GUR in general, in addition to all the chapters in this book, we suggest the following:

- Article from industry and research: Pagulayan et al. (2003), Thompson (2007), Kim et al. (2008), Isbister and Schaffer (2008), Mellon (2009), Drachen and Canossa (2011) and Zoeller (2010).
- Conference presentations: In general, there have, for the past 5–6 years, been a series of excellent talks at the yearly Game Developers Conference on the topic of game telemetry analysis, both in the main conference and the associated summits, notably the Social Games Summits.

- Websites: Another source of information is Gamasutra.com, whose features-section has been host to several GUR-oriented articles by some of the best people in the industry working in the area.
- The GUR-SIG: the International Game Developers Association hosts a Special Interest Group on Game User Research, which hosts a collection of GUR-literature and –references on the website: <http://www.mendeley.com/groups/758231/gur-sig/> (the collection may be moved in the future).

About the Authors

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Alessandro Canossa, Ph.D. is Associate Professor in the College of Arts, Media and Design at Northeastern University, he obtained a MA in Science of Communication from the University of Turin in 1999 and in 2009 he received his PhD from The Danish Design School and The Royal Danish Academy of Fine Arts, School of Architecture. His doctoral research was carried out in collaboration with Io Interactive, a Square Enix game development studio, and it focused on user-centric design methods and approaches: prototypical player behaviors are described procedurally and from those profiles game environments emerge that are able to accommodate all of the possibilities for action. In the course of his research, Dr. Canossa has published more than 30 articles, book chapters, and journal contributions, he has presented at several international conferences including Future Play, GDC San Francisco, GDC Canada, Mindtrek, IFIP Interact, IEEE Conference on Computational Intelligence and Games, ACM Foundations of Digital Games and DiGRA discussing topics from game user research, HCI, game metrics analysis and player experience. He has also received a Best Paper award at the largest media conference in Northern Europe, MindTrek, in 2009. Beside his academic activities he still maintain contact with the industry: his work has been commented on and used by companies such as Ubisoft, Electronic Arts, Microsoft, and Square Enix. Within Square Enix he carries on an ongoing collaboration with IO Interactive, Crystal Dynamics and Beautiful Games Studio,

where he has worked on titles such as Kane & Lynch: Dead Men, Tomb Raider: Underworld and Kane & Lynch: Dog Days.

Janus Rau Møller Sørensen: User Research Manager at Crystal Dynamics and IO Interactive (Eidos/Square Enix), managing games user research activities in the US and Europe. Janus has a multidisciplinary academic background in Information Studies and Ethnography & Social Anthropology, and he has worked in the game industry since 2004 – first in Denmark at IO Interactive, and since 2009 in the US at Crystal Dynamics – on titles such as Hitman Absolution, Deus Ex: Human Revolution, Tomb Raider, and Lara Croft & the Guardian of Light. Janus’s chief work-objective is to help enable great game experiences by putting a deep understanding of real people at the center of design decision-making; seeing gamers as human beings, not just behaviors or numbers in a spreadsheet.

References

- Canossa, A., & Drachen, A. (2009). *Patterns of play: Play-personas in user-centered game development*. DIGRA. London: DIGRA Publishers.
- Canossa, A., Drachen, A., Sørensen, J. R. M. (2011). Arrgghh!!! – Blending quantitative and qualitative methods to detect player frustration. In *Proceedings of the 2011 foundations of digital games conference*. Bordeaux: ACM Publishers.
- DeRosa, P. (2007, August 7). Tracking player feedback to improve game design. *Gamasutra*. http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_php
- Drachen, A., & Canossa, A. (2009a). Towards gameplay analysis via gameplay metrics. In *Proceedings of the 13th MindTrek*. Tampere: ACM-SIGCHI Publishers.
- Drachen, A., & Canossa, A. (2009b). Analyzing spatial user behavior in computer games using geographic information systems. In *Proceedings of the 13th MindTrek*. Tampere: ACM-SIGCHI Publishers.
- Drachen, A., & Canossa, A. (in press). Evaluating motion. Spatial user behavior in virtual environments. In O. Stammaa, A. Lugmayr, H. Franssila, P. Näränen, & J. Vanhala (Eds.), *Special issue on ACM Academic MindTrek 2009: Everyday life in the ubiquitous era*. Inderscience Publishers. *International Journal of Arts and Technology (IJART)*, 4(3), 294–314. doi:10.1504/IJART.2011.041483 (2011 issue).
- Drachen, A., Canossa, A., & Yannakakis, G. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *Proceedings of the IEEE computational intelligence in games* (pp. 1–8). Milan: IEEE Publishers.
- Han, J., Kamber, M., & Pei, J. (2005). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco: Morgan Kaufman. Available from Amazon here: <http://www.amazon.com/Game-Usability-Advancing-Player-Experience/dp/0123744474>
- Kennerly, D. (2003, August 15). Better game design through data mining. *Gamasutra*. Available from: http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_php
- Kim, J. H., Gunn, D. V., Suhuh, E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). *Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems*. Florence: Computer–Human Interaction (CHI).
- King, D., & Chen, S. (2009). Metrics for social games. In *Proceedings of the social games summit*, San Francisco.

- Kuniavsky, M. (2003). *Observing the user experience: A practitioner's guide to user research*. San Francisco: Morgan Kaufman Publishers.
- Lameman, B. A., Seif El-Nasr, M., Drachen, A., Foster, W., Moura, D., & Aghabeigi, B. (2010). User studies – A strategy towards a successful industry-academic relationship. In *Proceedings of future play 2010* (pp. 1–9). Vancouver: ACM Publishers. doi:10.1145/1920778.1920798.
- Lewis-Ewans, B. (2012, April 24). Finding out what they think: A rough primer to user research, parts 1 and 2. *Gamasutra*. http://www.gamasutra.com/view/feature/169069/finding_out_what_they_think_a_.php
- Longley, P., Goodchild, M. F., Macquire, D., & Rhind, D. (2005). *Geographic information systems and science*. Chichester: Wiley.
- Mahlman, T., Drachen, A., Canossa, A., Togelius, J., Yannakakis, G. (2010, 18–21 August). Predicting player behavior in Tomb Raider: Underworld. In: *Proceedings of the 2010 IEEE symposium on computational intelligence and games, CIG'10* (pp. 178–185), Copenhagen.
- Mellon, L. (2009). *Applying metrics driven development to MMO costs and risks*. Versant Corporation. <http://www.maggotranch.com/gdc2004.ppt>
- Pagulayan, R. J., & Keeker, K. (2007). Measuring pleasure and fun: Playtesting. In C. Wilson (Ed.), *Handbook of formal and informal interaction design methods*. San Francisco: Morgan Kaufmann Publishers.
- Pagulayan, R. J., Keeker, K., Wixon, D., Romero, R. L., & Fuller, T. (2003). User-centered design in games. In *The HCI handbook*. Mahwah: Lawrence Erlbaum Associates.
- Seif El-Nasr, M., & Zammito, V. (2010). User experience research for sports games. In *GDC submit on games user research*.
- Silverman, D. (1993). *Interpreting qualitative data*. London: Sage Publications Ltd.
- Sterne, J. (2002). *Web metrics, proven methods for measuring web site success*. London: Wiley.
- Thompson, C. (2007). How Microsoft labs invented a new science of play. *Wired Magazine* 15(9). URL: http://www.wired.com/gaming/virtualworlds/magazine/15-09/ff_halo?currentPage=all
- Vercellis, C. (2009). *Business intelligence: Data mining and optimization for decision making*. Chichester: Wiley.
- Zoeller, G. (2010). Game development telemetry. *Presentation at the game developers conference*, San Francisco
- Zoeller, G. (2011). MMO rapid content iteration. *Presentation at GDC Online 2011*, San Francisco, CA. URL: <http://gdc.gulbsoft.org/>

Chapter 15

Interview with Aki Järvinen from Digital Chocolate

Alessandro Canossa

Digital Chocolate is a development studio focused on social and casual games for PCs, smartphones and consoles. The firm was founded in 2003 by Trip Hawkins; since then it has won more than ten awards including “Best Game Developer of the Year” award from both IGN and Mobile Entertainment. The most successful games developed are free to play and include micro transactions on virtual goods.

Aki Järvinen has a double role in Digital Chocolate: as creative director and competence manager. He is supervising game designers, providing feedback on existing projects; at the same time, he is also the executive producer, leading the development of a new social game to be launched late 2012. His background is in game design, academic research and software development.

Q: How do you employ game telemetry in Digital Chocolate?

Aki: Digital Chocolate operates in the free to play market; our games are free, but include micro transactions on virtual goods. Thus, it is vital for us to know how many players convert into paying users, how many of them come back to the game, and with what frequency. In order to gain this intelligence, we simply have to adopt metrics tracking. We need to track pretty much everything players do in the game in order to understand how we can measure and affect a certain behavior. In the free to play market, it is not possible to survive without making use of telemetry systems, tracking key performance indicators and adopting metrics-driven development practices; for that reason we tend to track every player action. It is the product managers that analyze this data, detect patterns and extract meaningful insights for each game.

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

Q: What kind of features do you track in your games?

Aki: There is a set of common features that we want to track in all games, but then each product also has a set of metrics that make sense only for that particular game. The default features we track across all games are mostly related to player retention. For example, we measure metrics answering questions such as: out of all the players that join the game at day 1, how many will return at day 2 and subsequent days. Typically the most interesting initial periods are day 1, day 7 and day 30.

A second set of interesting metrics revolves around monetization. Such metrics are concerned with questions such as: how many players buy virtual goods, how much virtual currency they buy, what do they buy, and for how long.

The third default set of metrics concerns player acquisition. These are concerned with: how many new players come into the game, at what point in the game's life-cycle, how many people complete the tutorial and so on.

Q: Do you also track variables relating to user experience beside retention, monetization and acquisition?

Aki: We do track session length, when players log in during the day and how often (per day, per week and per month), but we also adopt a more qualitative paradigm for user experience testing before game launch. We generally try to nail the user experience before launch, because of the intrinsically softer nature of the problem. After that we concentrate on more quantitative metrics.

Q: Can you talk about the analytical practices in place at Digital Chocolate?

Aki: We have built an internal tool for analyses, visualization and reporting. The tool collects all the data that we track and presents it in pre-made formats according to who is viewing it. The tool helps us mine data and find specific answers to the questions we have in terms of acquisition, retention and monetization.

Q: Who are the stakeholders for your telemetry data?

Aki: Every game project has a product owner, the executive producer, who is the main point of contact. He is responsible for the business of the game, the schedule, the budget and all the rest. Besides that, each team also has a product manager who performs advanced analyses and reports to the product owner and together they report to corporate management. The corporate team has the overview of the whole portfolio, and makes ultimate strategic decisions; they receive reports from the telemetry system once a week, while executive producers and product managers work with the data daily. But in general the whole team has access to the metric tool; it's used by game designers and level designers alike, since their job is deeply influenced by how players use the game.

Q: Can you talk about the tool you use to gather and analyze data?

Aki: Well, it's not exactly rocket science; it resembles closely several third party solutions available off-the-shelf. We have chosen to adopt Tableau, KISSmetrics and Kontagent, and build from there. The reporting tools are dynamic, and different stakeholders can query the data and obtain personalized reports. At its present state the tool answers all the needs we have since it depends on the games we are developing at the moment, but that might change when we start making different types of

games. We have a team constantly maintaining and evolving the tool, so we can be ready if a certain game team asks for different features. In the end the teams are the customers that the tool is developed for.

Q: What advice would you give developers who are just starting now to use telemetry systems?

Aki: The key advice I can give you is that you can track all the data in the world, but if you don't know how to analyze it, if you don't understand what it means and how it is relevant, the data itself is worth nothing. It is important not to get infatuated with metrics, but try to use them in a pragmatic and informative manner. The way I see it is that metrics cannot help developing new, interesting features, but can be precious in evaluating features. You cannot expect metrics to magically fix your product automatically, but it can be helpful in optimizing existing features and addressing problems. It is not a straightforward process; it is till necessary to apply a creative problem-solving mindset.

Q: It is clear that you use metrics largely to verify hypotheses, for example how many people turn into paying customers, but do you also perform exploratory analyses, looking for meaningful patterns of use?

Aki: We would really like to do more of it, but it comes down to the fact that we don't really have the resources, the manpower or the time to do that. We have the data and the tools, but for now it's hard to justify the resources necessary for exploratory type of analyses. Having been in this business for years, experience has taught us which metrics yield the highest payback in terms of actionable information versus resources spent. We don't have yet the luxury to have our own research department, but in an ideal world it would definitely be very interesting and important. For example, right now we only focus on data on a per-product basis, but it would be great to look at data on a per-customer base, across different games, and for a longer period. This is where I see our future efforts being focused on.

Chapter 16

Better Game Experience Through Game Metrics: A Rally Videogame Case Study

Pietro Guardini and Paolo Maninetti

Take Away Points:

In this chapter, we present:

- How game metrics were designed, implemented and employed during the development of a rally racing videogame, as crucial support for user testing.
- How the game design took advantage of the issues identified by metrics.
- How reward systems such as Achievements and Trophies can be designed to work as game telemetry.
- Some general recommendations on games user research.

16.1 User Testing in Milestone Studio

Since its establishment in 1994 under the name Graffiti, Milestone maintained the title of flagship company of Italian videogame development – a rather small industry when compared to other European countries. Milestone’s developers have specialized in racing games, evolving from the first very successful arcade titles such as the Screamer series, to the realistic Superbike World Championship titles – SBK 2001 (2000), published by EA, is widely considered as the best bike simulation game. With the significant broadening of the video games audience in recent years

P. Guardini (✉)

Milestone S.r.l, via Gustavo Fara 35, 20124 Milan, Italy
e-mail: pietro.guardini@milestone.it; pietro.guardini@gmail.com

P. Maninetti

Milestone S.r.l, Milan, Italy

PopCap Games International, The Academy, Dublin 2, Ireland

and the resulting changes in the industry, Milestone acknowledged the importance of user studies to better understand its new, heterogeneous audience in order to improve the design and better target their games. As in the cases of many other developers (cf. Collins 1997; Fulton 2002; Fulton and Medlock 2003; Laitinen 2005; Davis et al. 2005; Thompson 2007; Greenwood-Ericksen et al. 2010), hosting voluntary participants in their Milan offices, asking them to play games and collecting their opinions became common practice: at first feedback was collected using interviews and questionnaires only, then video recordings, and later game metrics, were introduced.

Automated collection of quantitative data on user behavior has been widely discussed and employed for some considerable time by Human Computer Interaction practitioners and researchers (for a survey, see Hilbert and Redmiles 2000). With this type of data, it is possible to track in detail and analyze sequences of interactions between user and interface, in order to identify application usage and usability issues. Recently, the game industry has embraced this approach and applied it with great advantages.

The case study of the rally racing game *WRC: FIA World Rally Championship* (Black Bean Games 2010) was detailed in this chapter, explaining how game metrics were employed during the development of the title as a crucial support for user testing, as well as after the game shipped, in order to collect information on game usage which proved to be a precious resource for developers in the design of its sequel, *WRC2: FIA World Rally Championship* (Black Bean Games 2011).

16.2 Introduction to Rally Motorsport

Rally racing is probably the oldest branch of motorsport, dating back to the origins of motor competitions in the late nineteenth century in Europe. Unlike track-based competitions, rallies take place on normal roads in a point-to-point format, alternating transfer and competitive sections. Participants must leave the service park and reach a predetermined start point within a certain time, wait in a row with the others for a staggered start, then race to the finish point in the shortest time possible, and then finally drive to the starting point of the next competitive section. The sum of all competitive stages' completion times will dictate the final standings. Each rally hosts a total of 15–30 of such sections, called “special stages”, in a time period between 3 and 4 days.

Each year, the FIA World Rally Championship (WRC) calendar features 13 events, each of which takes place in a different country around the world. The type of road terrain drivers will come across strictly depends on the hosting country: in Sweden roads are covered with ice and snow, in the United Kingdom road surface is a mix of tarmac and mud, in Mexico drivers will find mostly gravel. Figure 16.1 portrays some of the environments' variety: reference photographs taken on actual special stages (left) are used by artists and designers to craft their in-game counterparts (right).



Fig. 16.1 Photographs taken on-site in Sweden, Jordan and Finland are used as references and for inspiration by rally stage artists and designers (Image courtesy of Milestone S.r.l. & Black Bean Games)

Track-based competitions require the driver to learn the circuit and memorize each turn lap after lap. In rally racing, track memorization becomes impossible, since the amount of road to race through is massive.¹ This is why the rally driver is supported by a co-driver who provides him with important information on the upcoming stretch of the road. These indications, called “pacenotes”, are needed not only to take the correct route at junctions, but also to make the driver aware of dangerous elements, which can affect fast driving. For example: the severity of corners, variations of the road surface’s tilt angle, the presence of hill crests – which will lead to jumps, changes in road surface, irregularity on the road, and so on. It is very important that pacenotes are conveyed with an appropriate rhythm and sufficiently in advance, in order to let the driver adjust car trajectory and speed. That is why pacenotes are written and then read out loud during the competition using

¹ In fact, there is an old saying between rally drivers that goes, “Circuit racers see 10 turns 1000 times while rally drivers see 1000 turns 1 time!”

specific abbreviations and codes. For example, the severity of corners is conveyed using a number from one to six, one corresponding to hairpins and six corresponding to wide-open corners. Several car manufacturers participate in rally events with specifically modified production cars: power and sheer speed are important, but the ability to adapt the car to different road conditions is very important as well. These are, in summary, the defining aspects of rallying.

16.3 Definition of Designers' Intents

The designers' main goal was thus formulated during the pre-production of *WRC: FIA World Rally Championship* (Black Bean Games 2010): to transfer all the aforementioned defining aspects of rally racing into the video game in order to craft an authentic and engaging game experience. The first and most important aspect was deciding which type of car control would be implemented in the game. Car control stems from the interactions between the player's activation of the control device and the car handling physics model. This is a dedicated part of the physics engine – the part of the game that embeds the rules governing how entities within the game environment interact (Hecker 2000). In a driving game, the handling model defines the game itself: by specifying how simplified or realistic it should be, gameplay is directly affected, making the game easier or harder to approach, simpler or more demanding or to master.

The designers' intent was to create a “realistic but approachable” car handling model. This definition implies two concepts, only apparently in contradiction. A high degree of realism makes the game more complex to play, since it implies that the handling model has to reproduce – and thus the players have to react to – a great number of factors affecting car behavior. If such complexity is reproduced, players familiar with rally motorsport and TV rally broadcasts will find themselves at home, being able to readily recognize these behaviors, transfer them and find confirmation to their previous knowledge (Bogost 2011). However, such complexity should not discourage less rally-experienced players: they should also be able to approach the game and have fun without the need to possess pre-existing skills (Guardini 2002). This is why designers decided to implement three different driving aids – braking assistance, traction control and trajectory help – to support less experienced players in, respectively, controlling the speed, avoiding wheel spins on slippery surfaces, and limiting skidding around bends. A few minutes should be sufficient to pick up the relevant skills when driving aids are activated. Great attention has been paid to the design of the special stages as well: the roads that players will race through have to provide great variety and they are responsible for the level of challenge offered to the players.

The presence of other contestants in a race can indeed dramatically change the perception of a game and the reported levels of fun, since beating opponents is an intrinsic positive reward (cf. Hopson 2001; Clark 2010). Having a direct point of reference to the player's performance is also crucial to assess her or his ability,



Fig. 16.2 A screenshot from the Rally of Sweden. The in-game interface includes, from *left to right* clockwise: (1) the *progress bar* which indicates the position of the player's car relative to the length of the current stage; mirroring the stage, the bar is split into five sectors by checkpoints; each sector can be colored in either *red* or *green* depending on the player's performance; (2) pace notes appear in the *middle* of the screen, as a visual support to the spoken ones; the current note indicates an upcoming jump; (3) stage time: low times are set by skilled players; (4) speedometer, rev counter and gears indicator; (5) damage reported by the car to specific components (Image courtesy of Milestone S.r.l. & Black Bean Games)

and it also contributes to teaching the players how to play – i.e., players get to know how to play by observing the computer controlled opponents as seen in Griffiths (2009). However, since WRC games have to faithfully portray the real WRC experience, the players race against the clock, without opponents to overtake. Their performance, defined by their times, is compared to the times set by the computer-controlled opponents at specific checkpoints during each stage and at the finishing line.²

Other intents have been reviewed and included in the game design document, detailing different game modes (such as progression in single player career) and game contents (such as unlockable official cars). For the purposes of this chapter, two important aspects will be discussed: car handling and special stages design.

Figure 16.2 illustrates the game interface and how designers decided to convey all the relevant information to the player: the progress bar on the left indicates (a) the position of the player's car relative to the length of the actual stage, and (b) the performance of the player in each of five sectors in comparison to the previous one (green=better, red=worse); pacenotes appear in the middle of the screen; the race time, speed, gear and car's damage are displayed on the right.

²Early in development, computer-controlled opponents are not appropriately tuned (i.e., opponents drive too slow or too fast, being too easy or impossible to beat) or even absent. In these situations, games user researchers must be very careful in assessing the game and always provide a context, especially when dealing with subjective ratings of game difficulty and fun.

16.4 Testing Designers' Intents with Players

Once designers have established their intents and a working prototype of the game becomes available, a series of playtest sessions are set up in order to test these intents with actual players. At Milestone, user testing studies are usually carried out by inviting voluntary participants to the offices based in Milan. Participants are then asked to play one or more games while the games user researcher collects both gameplay data and players' opinions on specific features of the game being tested. The first type of data, as described in various chapters in this book, is telemetry data, which describes what players do within the game and how they interact with the game system. They are collected automatically during the gameplay sessions with audio-video recordings and game telemetry systems. They are called objective (or behavioral) data since they accurately describe the players' behavior. In contrast, the second type of data describes what players think and feel about the game and its features, whether they find it fun and what they appreciate the most and the least. They are collected by directly asking questions to players using interviews or questionnaires. Since this type of data is produced by an individual after actively thinking about the issue, they are called subjective (or attitudinal) data.

As argued for in Chap. 14, only the integration of objective and subjective data can give a complete picture of the players' experience with the game (cf. Fulton and Medlock 2003; Ambinder 2009). Having two distinct types of feedback is important since objective data describe the actions and give insights about players' intentions, but they do not reveal anything about players' motivations. By taking into account both of them, it is possible to correlate the chain of actions to reported levels of fun, and find out which moments in the game are judged as fun and involving or on the contrary repetitive or dull. For example, a successful technique consists in embedding into the game a brief survey that players are requested to answer every 3 min by selecting the appropriate answer with the joypad – thus without putting down the controller to minimize distractions from the game itself (Schuh et al. 2008; Amaya et al. 2008). A similar mixed-method approach is discussed in Chaps. 21 and 22. It should be noted that subjective data can be prone to biases due to players' expectations and the particular situation of playing a new game in the developer's offices: an accurate methodology and the comparison of feedback from different sources can also help in controlling and minimizing the problem.

Testing car handling and special stages design requires collecting data in order to answer the following questions:

- Do players find the car handling realistic?
- Do players find the car handling approachable?
- Do players find the stage design challenging?

Collecting subjective data is, in this case, straightforward: the researcher has to directly ask players these questions during individual interviews or employing a

questionnaire. It is possible to use open answers, where the respondents are free to answer in the preferred way, or to provide respondents with closed answers or a scale on which they can base their answers. With the latter solution, the answers are standardized and ready to be compared across respondents and gameplay sessions.

As previously discussed, simply asking players these questions – hence collecting subjective data – is not enough, however. It is also necessary to answer the questions by examining the unbiased data about players' behavior – that is, objective data collection is needed. Two data sources were at disposal: audio-video recordings and game metrics. In the first case, a video stream records what is displayed on the TV screen connected to the game console or to the PC monitor. If the study also involves manual interaction and facial expressions, additional video streams will record the movements of the hands on the controller and the face of the player (or players in case of multiplayer game testing). In the second case, a specific piece of software is employed to collect data directly from the game environment. Every event is logged in a specific file or database: activations of the game controller, the corresponding actions performed by the player's avatar, as well as the consequences to those actions; events related to the computer-controlled agents are also recorded.

There are other objective data metrics available in the field of games user research: biometrics. Instead of logging changes in the game system, they measure changes within the player's body during the gameplay sessions. It is, therefore, possible to measure the valence and intensity of players' emotional states and the patterns of their eye movements on the TV screen. This has been made possible with the application of methods from psychophysiology – the branch of psychology that studies the physiological bases of psychological processes. Their successful implementation in product design and evaluation in the discipline of human factors (see Picard 1997) has convinced several games user researchers to explore this new ground (see chapter 13 and 14 in Isbister and Schaffer 2008; Van der Heijden 2010; Ambinder 2011; Zammito 2011).

16.4.1 Considerations About Audio-Video Recordings

Audio-video recordings of gameplay sessions are perhaps the easiest way of collecting objective data about the players' behavior. Every detail is “on tape” and ready to be examined at any time by the games user researcher looking for significant events. The format is also ready for immediate fruition: there is no need for the viewer to know anything about data visualization and statistics. However, this method is not the most efficient with regards to processing time. Moreover, its richness in details can be overwhelming. For example, in order to compare different sessions, it is necessary to convert in numerical form all the recorded events. This implies assigning a score to every significant event that happened in each session, and then to compare the respective scores. Moreover, if the scoring procedure is carried out by several researchers (in order to speed up the process), it may be prone to subjective biases: for example, in a racing gameplay session what one judge

assesses as a crash with severity “4” on a 5-point scale, may result in a “3” or even “2” crash by a second judge who supposedly is using the same criteria. In other words, objective data become subjective when they are converted into numbers by a human observer. This issue is well known in observational research and is kept under control by raising the number of observers (“judges”) and by employing very well defined criteria for scoring.

Audio-video recordings are currently employed in Milestone’s user research activities as a support, when specific issues need a deeper investigation with the involvement of game designers. While players usually can’t explain in detail what happened during (for example) an unexpected crash – a common answer is “I don’t know... something strange happened”, designers are well aware of game mechanics and they are more likely to understand how and why some events happened by watching the video recordings.

In the future, automatic and foolproof event loggers will be at the disposal of user researcher, thus solving the time-consuming and subjective-scoring problems of video analysis. However at this time, a specific game metrics data logger is a much more flexible and complete solution in analyzing and understanding the objective interactions between players and games.

Audio-video recordings will not be mentioned in the rest of this chapter, although they are always present and employed in Milestone’s user research activity in the aforementioned case.

16.4.2 Experimental Design Comes First

Before delving into the description of the design and implementation of the game metrics logging system employed in this case study, it is crucial to point out the importance of an accurate planning of the test. User testing activities are most often considered to be true scientific experiments, with the goal of putting specific hypotheses to the test (cf. Speyrer and Jacobson 2006; the interview with Randy Pagulayan in Isbister 2006; Parker 2011). These hypotheses always represent either plausible outcomes of an action or possible explanations for an event, and in both cases they can be confuted or validated exclusively by testing them against the data collected from user testing sessions. Therefore, only when the hypotheses are clearly specified, it is then possible to design the experiment and choose the appropriate methods and tools, which will guarantee a correct collection of data.

Having clear hypotheses from the beginning of the study is mandatory, when doing hypotheses driven testing. (Note: sometimes exploratory based analysis is performed on data, as explained in Chap. 13, we are mostly discussing hypothesis driven analysis here). The alternative “let’s record everything and then we will see” approach should be avoided at all costs. While on the one hand it is tempting to take into consideration every existing variable and log them all in order not to miss something important – on the other hand it is very likely to be overwhelmed by the consequent enormous amount of data at the stage of analysis, with the result of

having to invest an unnecessarily long time in performing many possible comparisons between variables in search for significant results. More practical problems involving software and hardware limitation, as well as data storage and management are also to be considered.

Moreover, when designing and running an experiment (or more generally any information collection activity), it is necessary to follow a series of specific procedures in order to be sure that the data are correctly collected, and that the conclusions drawn from their analysis are valid. There are in fact several potential sources of error to be aware of.

The order of presentation plays an important role, as in the following example. A number of participants were asked to play four rally stages in a row and consequently rate their experience in order to assess stages' difficulty and appeal. Results showed they had rated as easier and duller the last stage in comparison to the first ones. But since participants learn how to play as their experience with the game grows, it is not possible to determine whether this result is due to an actual difference in the stage design, or otherwise to an improvement in players' skills acquired during the four sessions, or even to a drop in participants' motivation, or just to a fatigue issue. These are the possible consequences of the order in which the stages are presented to players, and something that researchers would want to control. Control of presentation order is achieved by employing the technique called counterbalance, which involves changing the order of presentation of the four stages for each participant.

This is just one issue researchers have to deal with in order to meet valid conclusions. A great number of publications on experimental research methodology are available, many have been published specifically on user experience measurement (Kuniavsky 2003; Tullis and Albert 2008; Schumacher 2010), and several of them deal precisely with videogames user testing (Collins 1997; chapter 11 in Blythe et al. 2003; Isbister and Schaffer 2008; chapter 9 in Fullerton 2008; chapter 25 in Schell 2008; Bernhaupt 2010).

In order to design an experiment which can effectively contribute to the quality of the game, it is also necessary to *operationalize* the concepts involved in the study. Operationalization is the process of defining general concepts in a clear and measurable way. For simple and clearly defined concepts this procedure is straightforward, while for more abstract concepts the process requires some assumptions.

In this case study, it was necessary to answer the following designer-intent related questions:

- Do players find the car handling realistic?
- Do players find the car handling approachable?
- Do players find the stage design challenging?

Realism and approachability of the car handling model and challenging stages design are very broad and general concepts, which at a first glance appear to be poorly adaptable to measurements and verification through objective data. It is indeed necessary to operationalize these concepts.

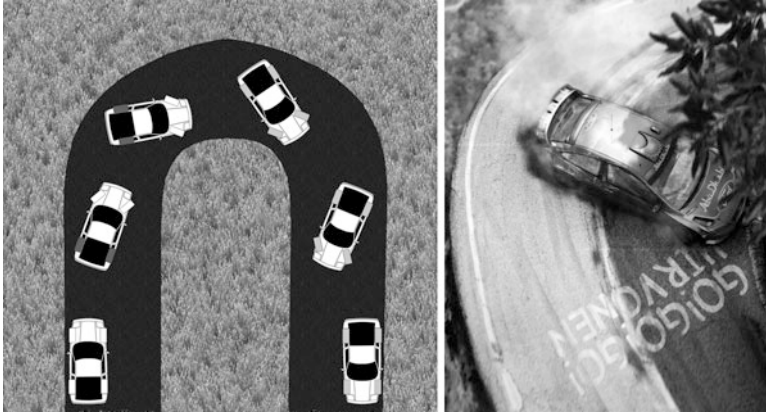


Fig. 16.3 Drifting implies traveling through tight corners in over steering, the rear wheels without traction, and the front wheels pointing in the opposite direction to the turn. *Left*: a scheme of how drifting is achieved. *White tires* indicate deceleration; *dark grey tires* indicate hand-braking; *light grey tires* indicate acceleration (for additional rally maneuvers cf. Sanches 2008). *Right*: a WRC artwork, depicting a car drifting through a corner (Image courtesy of Milestone S.r.l. & Black Bean Games)

- **Realism:** a realistic car handling model should allow players to drive in “rally style”, as they have seen on TV. Therefore, it was assumed that maneuvers such as drifting, handbrake turning, corner cutting, jumps, and high-speed driving would be found during gameplay sessions. Expert rally-videogame players may contribute to a greater extent here in comparison to players who do not know about rally racing and are likely to drive in an unspecialized way. In fact, expert players should be able to transfer the knowledge they gathered by playing other rally games to new ones. Even real-life rally drivers with familiarity with rally videogames may participate in the study. Figure 16.3 illustrates one of the most common rally maneuvers: drifting.
- **Approachability:** an approachable car handling model should allow all types of players (expert and novice) to understand how to drive their vehicle within a few minutes. The first experience with the game should not be problematic. In general, particular attention should be paid to the very first minutes participants spend with the game, which are indeed crucial for capturing the player’s interest, and persuading her or him to keep on playing (Fulton and Romero 2004). Game learnability is further investigated by examining what happens in the following moments: improvements should be found between the very first minutes of gaming and the following ones. These improvements should be found when comparing stage completion times, driving accuracy, and number of collisions with roadside objects (see Fig. 16.4). Learning curves should also provide information on the general difficulty for each rally stage.
- **Challenging:** a challenging stage design should provide players with enough variety throughout each single stage, as well as variety between different stages. Therefore, it was assumed that differences would be found in the way players race



Fig. 16.4 Loss of control may cause collisions with roadside elements such as fences. As a consequence, the car can suffer damage to specific mechanical parts such as engine, brakes, etc. The damage can affect car performances (Image courtesy of Milestone S.r.l. & Black Bean Games)

Table 16.1 This table illustrates which type of data (rows) have been considered in this case study in order to test designers’ intents (columns)

Designers’ intents to test	Realistic car handling	Approachable car handling	Challenging stage design
Subjective data	Interview, questionnaire	Interview, questionnaire	Interview, questionnaire
Objective data	Expected rally maneuvers: drifting, handbrake turning, cutting corners, jumps, high speed Expected better performance by expert rally videogame players in comparison to novices	Expected improvements in completion times, driving accuracy, number and entity of collisions	Expected differences between frequencies of controller activation

(a) between different sectors of the same stage and (b) between different stages. These differences could be found by comparing frequencies of controller activations: the use of gas, brakes, steering and handbrake are assumed to be reflecting the overall configuration of the stages, as well as the different types of terrain.

Table 16.1 summarizes which types of data are to be collected in order to test designers’ intents.

It should be noted that the required assumptions made here during operationalization are not unique and universally valid. Different assumptions have to be made when testing videogames belonging to different genres, and even when testing the same type of videogame it is possible to choose differently. For example, in an

interesting study on game design quantification, Ip and Jacobs (2004) assumed that by measuring car acceleration and braking times, it was possible to draw conclusions on how realistic the car handling model was perceived to be by players across ten different rally games. Their hypothesis was that shorter acceleration and braking times were more likely to be found in less realistic and more pick-up-and-play games. Their data confirmed this assumption.

Before finalizing the assumptions and proceeding with data collection, it is advisable to discuss these assumptions with the game designers and collect their opinions. Ideally, designers should provide researchers with already operationalized hypotheses, ready to be tested. In two distinct publications (Schuh et al. 2008; Romero 2008), the Microsoft Game Studios user research team detailed how designer intents were operationalized and tested in the “Time Trials” game mode of the racing title *Forza Motorsport 2* (Microsoft 2007). In their case, the intent was that that particular game mode had to be challenging but not overly frustrating. The designers themselves proposed from the beginning a clear hypothesis to test: “approximately 80% of the target users should be able to complete any particular time trial (...) after ten laps” (Schuh et al. 2008; p. 252). However, the design team does not always have the time to produce these explicit statements.

Another important issue is planning in advance which comparisons are to be made in order to test the designers’ intents and reply to the research questions. For example, the hypothesis about car handling realism involves comparing the driving style of expert and novice rally videogame players. Typical rally maneuvers are expected to be present to a greater extent among expert gamers: analyzing and comparing data from the two types of gamers will reveal whether this hypothesis is true, and to which extent. In the case designers provided themselves a target value to test (for example: expert players should drift 50% more than novice players in a particular stage), this will be directly compared to the data collected from both expert and novice players.

In order to test the second hypothesis on car handling accessibility and learnability, it is necessary to verify whether the number of errors such as collisions with roadside objects and off-road departures decreases with playing. In fact, as previously noted, players learn how to play as their experience with the game grows. Therefore, it is reasonable to expect an improvement when comparing performances at the beginning of a stage with the performance at the end, or comparing performances between the first and the fifth stage raced, for example. To achieve this, each rally stage is divided into consecutive sectors, and a series of comparisons are then performed. For example, by comparing the data relative to the first sector with the data from the second sector, it is possible to observe whether an improvement in car control is present (for example, less collisions), or whether such an improvement is found further on. Moreover, by comparing these indexes between different stages and different rallies it is possible to draw conclusions on their relative difficulty and trace a graph depicting the learning curve that characterizes their difficulty for each group of participants, novices and experts.

The learning curve is extremely helpful. This notion makes it possible to verify whether the intended difficulty specified by the designers is the actual difficulty

experienced by players in each portion of the game: at the beginning, thus providing insights on stage approachability, but also across the entire stages. McAllister and White (2010) mention how the developers of the off-road racing title *Pure* (Disney Interactive Studios 2008) were able to tune the difficulty of the game with successful results by adjusting the dynamic A.I. system to the learning curve plotted from performance data collected with user testing sessions. Moreover, providing designers with a difficulty score for every stage is extremely helpful when they have to create game modes with a sequence of events such as the career, which is generally identified as the core of every racing game (Pagulayan et al. 2003). Indeed, it is advisable to avoid placing a very difficult rally stage at the beginning of the career mode, when players are not acquainted with the controls and the handling model.

The procedure of fragmenting the entire game into different sectors and then comparing users' performance relative to these sectors has been vastly employed in games user testing. The application of this procedure is straightforward with games structured in levels or missions from the beginning. For example, Schuh et al. (2008) described how during the testing of *Halo 2* (Microsoft Game Studios 2004), the popular first person shooter game, the number of player deaths were compared between not only missions, but also between subsequent enemy encounters.

The summarizing table can now be updated with the planned comparisons discussed in this paragraph (Table 16.2).

As with any other product, user testing games is an iterative process. Testing designers' intents only once during the development process is not enough. Game development is a long and complex process that may last between 18 and 24 months in small teams (McAllister and White 2010), and up to 3 or 4 years for bigger productions (Bethke 2003). Numerous changes are being made in gameplay tuning and entire game features may even be deleted or introduced during development: user testing has to continuously monitor the game to be effective. Here, the three aspects under examination – realism and approachability of car handling, and stage design – have been tested throughout the entire development process: starting as soon as the handling model was sketched and the first rally stages became available, until the lack of further time and resources make it impossible to actually apply user testing results. Once the designers' intents have been transformed into directly measurable hypotheses and it is clear which comparisons are to be performed, it is possible to proceed with the implementation of the game metrics collector.

16.5 Game Metrics Collector Design and Implementation

Within a racing videogame, hundreds of parameters change their values every second and most of them even faster. At every frame, the scene is updated by the rendering engine according to the outcome of the interactions between the input provided by the player through control-device activations and the corresponding consequences dictated by the physics engine (Hecker 2000). For example, when the player drives around a bend, the input from the stick assigned to the steering

Table 16.2 This updated table illustrates which type of data (rows) and planned comparisons have been considered in this case study in order to test designers' intents (columns)

Designers' intents to test	Realistic car handling	Approachable car handling	Challenging stage design
Subjective data	Interview, questionnaire	Interview, questionnaire	Interview, questionnaire
Objective data	Expected rally maneuvers: drifting, handbrake turning, cutting corners, jumps, high speed Expected better performance by expert rally videogame players in comparison to novices	Expected improvements in completion times, driving accuracy, number and entity of collisions	Expected differences between frequencies of controller activation
Planned comparisons	Expert vs. novice rally videogames players Designers' operationalized intents about rally style racing vs. collected data (when available)	First sector vs. subsequent sectors of different stages Driving aids on vs. driving aids off Expert vs. novice rally videogame players Designers' operationalized intents about expected players' performances vs. collected data (when available)	Fast vs. slow stages Tarmac vs. gravel vs. snow Designers' operationalized intents about expected players' performances vs. collected data (when available)

command is sent through specialized filters before reaching the steering wheel of the virtual car. Here, the physics engine applies the movements, with regards to the joints and constraints of the suspensions geometry. It also calculates the final output considering car position, acceleration, weight and speed, the grip level relative to each tire, terrain conformation, and other factors.

Between these numerous indexes, it is necessary to identify the ones that actually convey significant information in relation to the research questions specified above. These will be tracked by the automatic data collector software and logged for analysis.

16.5.1 Which Metrics Were Collected

The position of the car at every moment is highly informative, since using this variable it is possible to recreate the trajectories that each player decides to follow while racing. This index will also show when the player is having trouble in keeping the car on the road, as well as when corners are cut and when the car “respawns” – i.e., it is repositioned in the center of the road. Respawning can happen either by manual control or automatically: in the first case, the player activates the respawn after an off-road departure because the car got stuck or it is not clear to him which direction to go, while in the second case, the respawn is triggered when the car is being driven too far away from the road in order to prevent excessive road cuts and thus cheating. In both cases, car respawning signals that something wrong has happened and the event needs further investigations.

The position of the car is tracked in two distinct ways: by storing its three spatial coordinates XYZ relative to the game environment, and by specifying the distance of the car from the end of the stage, in meters and percentage of completion. While the first measurement provides detailed information, which can be compared between different participants, only the latter allows a direct comparison between stages.

Information about where collisions occur along the stages and the resulting amount of car damage is useful to identify potential problems in stage design and in-game interface effectiveness: forthcoming tight bends or obstacles which require hard braking may be signaled in an ineffective way, or the roadway design may be tricky. Collisions are tracked by a collision detector, which is activated in case the distance between the car and another object becomes zero.

Drifting is an important aspect of racing on loose terrains, such as gravel and mud. This technique consists in driving through a bend by putting the car in oversteer and consequently losing rear wheel traction while maintaining vehicle control and high speed. The main advantage is that bends are travelled faster, since the orientation of the car and its wheels makes it possible to accelerate very early out of the turn. A specific drifting index was calculated and tracked by logging the value of the angle between the direction of the car and the direction of its speed. The left section of Fig. 16.5 explains how the drifting index is calculated.

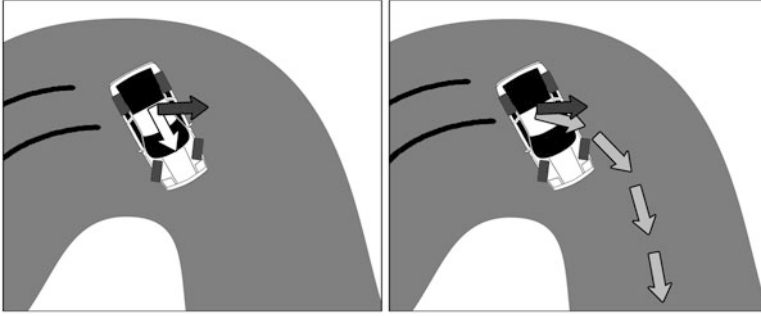


Fig. 16.5 The drifting index (*left*) takes into account the angle between the direction of the vehicle (*white arrow*) and its speed (*dark grey arrow*). The driving accuracy index (*right*) takes into account the direction of the speed of the vehicle (*dark grey arrow*) and the direction of the road (*light grey arrows*)

Handbrake turning is another technique used in rally racing to go through very tight corners as fast as possible. After turning the wheel in the direction of the turn, the driver lifts the handbrake and thus locks the rear wheels. As a result, the car starts to drift and slides sideways, thus making it faster to drive through corners, such as hairpin turns. Handbrake activation is performed by pressing a specific button on the controller.

Controller activation frequency is a raw index of how demanding on the controls the driving is. Every movement of the sticks, triggers, and buttons provides information on activations and modulations of the car controls. A fast stage with many long straight segments and wide corners will require less control operation than a stage with many tight bends and hairpin turns. This information will help to characterize the complexity of each stage: just think about the differences between a relaxing drive along a wide coastal road in comparison to driving on a tortuous road through mountains. Furthermore, it is possible to identify which part of a stage is the most demanding by splitting it into several sectors and comparing the frequency data relative to each single sector, in the way previously described in this paragraph.

It also seems reasonable to assume that if a novice player is having trouble in keeping the car on the road, he will continuously adjust the car's trajectory by steering and counter steering, by acting on the brakes and modulating the gas. Consequently, this will result in high frequency values of activations of the corresponding control. On the other hand, an expert rally player will act on the controls to the minimum extent needed in order to effectively follow the road. This pattern of interaction has been found during the analysis of metrics relative to other Milestone titles.

An accuracy driving index was also necessary: in order to track how good a driver each player was, it was decided to employ the value of the angle between the direction of the speed of the vehicle and the direction of the road. This way, accuracy is tracked whether the car is drifting or not, and independently from the trajectory chosen by the player, by producing an angular displacement between the correct and

the actual driving lines. The right section of Fig. 16.5 explains how the accuracy index is calculated.

Average car speed and stage completion times both provide useful information about how fast a single stage is in comparison to other stages in the same rally. A comparison can also be made between sectors within a stage, or between stages and sectors belonging to different rallies.

The summarizing table requires a new update (Table 16.3).

16.5.2 How Metrics Were Collected

In general, data collection can be triggered, or sampled, as discussed in Chap. 2. In the first case, a specific event within the system triggers the metrics collector, which in turn writes in a log file a row of previously specified data about the system environment and its context. In other words, sampled data collections is similar to taking a picture of the system in the specific moment the triggering event happened, taking into consideration both content and contextual information, such as time, date and GPS localization data. In the latter case, the data is continuously collected by saving a row of measurements from a number of indexes at regular intervals. Sampled data collection, therefore, is like recording a video: a series of pictures of the system and its context are saved, for example, once per second. This process is called “sampling” in signal processing theory, and the sampling rate specifies how many measurements are performed within 1 s. If pictures are taken more frequently, more details are collected, and it becomes less likely that a very fast event goes undetected. However, taking too many pictures is not recommendable, since their amount may become unmanageable at the analysis stage. Moreover, too much detail does not automatically translate into better feedback: a level of detail that goes beyond the one requested to answer the research questions is not useful, therefore not needed.

In this present case, choosing between the two methods was straightforward, since the genre of game under test required a continuous gameplay description. For example, data about the trajectory of the car are requested at all times and not just at certain points in the stage. When testing other genres, especially ones based on a low-pace gameplay, the employment of an event-based logging system is more adequate. In the case of card-based games, turn-based strategy games, or adventure games it is not necessary to log gameplay data with such detail.

To ensure maximum flexibility and to keep the amount of collected data under control, it was decided to link the sampling rate to each frame – so that each “picture” of the system would match what was being displayed on the screen – and to introduce the option of specifying its value by modifying a single line of code. In other words, it was possible to specify the frame interval between measurements: from one, corresponding to one row of data for each frame, to any given positive number. With a game running at 30 frames per second, collecting a

Table 16.3 This updated table illustrates: (a) which type of data, (b) planned comparisons, and (c) indexes have been taken into account in this case study in order to test each designers' intents

Designers' intents to test	Realistic car handling	Approachable car handling	Challenging stage design
Subjective data	Interview, questionnaire	Interview, questionnaire	Interview, questionnaire
Objective data	Expected rally maneuvers: drifting, handbrake turning, cutting corners, jumps, high speed Expected better performance by expert rally videogame players in comparison to novices	Expected improvements in completion times, driving accuracy, number and entity of collisions	Expected differences between frequencies of controller activation
Planned comparisons	Expert vs. novice rally videogames players Designers' operationalized intents about rally style racing vs. collected data (when available)	First sector vs. subsequent sectors of different stages Driving aids on vs. driving aids off Expert vs. novice rally videogame players Designers' operationalized intents about expected players' performances vs. collected data (when available)	Fast vs. slow stages Tarmac vs. gravel vs. snow Designers' operationalized intents about expected players' performances vs. collected data (when available)
Indexes to record	Position of the car in respect to the road Speed Drifting index Handbrake activations	Times Driving accuracy index Collision detection Car damage	Frequency of controller activation Position of the car in respect to the road Speed Drifting index Handbrake activations Times Driving accuracy index Collision detection Car damage

row of data each frame means collecting one line every 0.03 s; the result of a 4 min long rally stage completion will give rise to 7,200 rows of data, each including one column for every specified index to be logged. With a game running at 60 frames per second, the situation above will give rise to 14,400 rows.

Establishing which sampling rate to use involves considering the tradeoff between having a very detailed numerical representation of gameplay sessions and having to deal with a very large amount of data. As already mentioned, the sampling rate must be decided taking into account the type of gameplay under investigation and the type of controller employed for interacting with the game. A fighting game with frantic button-smashing combos will need a higher sampling rate, while on the contrary a chess simulator will need a much lower sampling rate, if not an event-based data collection system. A formula that may help in determining the sampling rate interval is the one known as Fitts' Law (Fitts 1954), which accurately models the human movement of pointing, and predicts the time required to move to a target as a function of the distance to the target and its size. Although this model was based on pointing, its accuracy holds for aiming movements, the most common within graphic user interfaces. Fitts' Law is a powerful tool for effective web design, and it has been employed to compare the performances of different input devices in 2D and 3D environments (for some gaming related studies, cf. Looser et al. 2005; Isokoski and Martin 2007; Natapov et al. 2009). However, even though the model has been extended to cover more complex tasks such as trajectory-following – as in steering a vehicle (Accot and Zhai 1997, 1999) – in the present case, it may be not reliable due to lack of constraints (it is acceptable to go off the road and the best trajectory never corresponds to the road axis). Therefore, a more practical approach was followed: the speed of the fastest player-generated event within the game system was measured, then divided by two, and the result used as a sampling rate. The fastest movement is usually the pressure of a button: here buttons are used to change gears and activate the handbrake. After some testing to analyze small steering corrections and handbrake activations, the sampling rate chosen was ten frames per second.

The implementation of the data collector was made directly into the “debug” version of the game, which is the work-in-progress, easily editable working version of the game that is normally used in parallel with the many “builds” required by the development process. The data was saved in a formatted text file, ready to be imported in programs such as Microsoft Excel (Microsoft 2010), SPSS (IBM 2011), and Tableau (Tableau Software 2003) for subsequent filtering, analysis, and visualization, respectively.

16.5.3 How the Data Was Analyzed

The importance of planning in advance has been emphasized more than once in this chapter: advantages are clear since results are delivered to the development team as early as possible, while careful planning translates into an increased flexibility to

cope with unexpected contingencies at any stage of the testing. In a similar manner, it is recommendable to think in advance, during the planning stage of the study, about how best to process and analyze the data. In particular, one must determine which type of statistics will be needed in order to test designers' intents and answer all the research questions (for a practical introduction, see Tullis and Albert 2008).

There are two groups of statistics to be considered: descriptive and inferential statistics (see Chap. 12). Descriptive statistics summarize the collected data set and tell the researcher in a quantitative way what happened during the gameplay sessions. They reveal how each single participant has performed, or how different groups of participants have performed by collapsing together single participants' data. On the other hand, inferential statistics let the researcher go beyond the small group of participants that actually was involved in the test, in order to draw conclusions on a much larger population, which includes all individuals with similar characteristics to the participants, such as age or gaming habits. Based on the collected data and taking into account the random variability of individuals' behavior, it is possible to generate statistical propositions with a certain degree of probability.

Descriptive statistics include measures of central tendency, such as mean and median, and measures of variability of data, e.g., standard deviation. Applied to a series of measurements collected from a group of participants, mean expresses the average value for the distribution, while median is the middle value of the distribution – half the participants achieved a score below the median, while the other half reported a score above it. Considering both values is important in the case of participants with extreme values, which may skew the mean towards one extreme or the other. Standard deviation represents how spread out are the data relative to the mean. Measures of variability are important since they tell whether participants perform a specific task in a similar manner (low variability) or they perform very differently (high variability), and to what extent. Moreover, the smaller the standard deviation is, the greater is the confidence with which it is possible to extend the findings to the wider population, that is, other gamers than the ones who participated in the user testing session.

Inferential statistics include:

- Confidence intervals, with which it is possible to estimate the range of values relative to the population on the basis of the values collected from actual participants;
- Curves estimation, with which it is possible to find a curve that graphically sums up the collected data;
- Hypothesis testing, a set of statistical tests through which it is possible to refute or accept a hypothesis on the basis of the collected data – mainly whether or not significant differences are present between distinct groups of participants.

The present study employed both types of statistics. Inferential statistics are more appealing, since they provide information that goes beyond the actual participants, making it possible to formulate predictions on other users and therefore on gamers and reviewers, although within the limits of probability. Nevertheless, only descriptive statistics can portray behavioral differences, thus being particularly useful to pinpoint specific problems. Raw data, i.e. the single scores each participant has obtained, can also be particularly informative in specific cases. For example, by

analyzing single scores instead of central tendency measures, it is possible to discover less frequent (or even unique) scores that may indicate problematic behaviors that otherwise will be hidden by mean values. When possible, it is advisable to examine the full spectrum of available data, from the results of inferential tests to single scores, in order to avoid leaving out potential issues – even if these concern just one participant and do not statistically represent the group. Because of the great variability of situations a player can come across in a videogame, it may be that only one user test participant has faced a particular situation that no one else has experienced, thus providing valuable feedback.

16.6 Participants' Selection

While following rigorous methodological procedures greatly contributes to ensuring the validity (and thus usefulness) of the collected feedback, much also relies on selecting the appropriate participants. Potential problems can be present at both the recruiting and scheduling phases (Bojko et al. 2010); however, the latter essentially refers to participants' no-shows and tardiness, while the first requires more attention. For example, the opinion of a non-gamer may be useful in testing a family game, while if asked about control responsiveness in a racing game he will likely find it difficult to answer, probably concluding that “the game is fine, it is just that I am not good at videogames” (cf. Snitker and Jeffers 2010). While potentially valuable at the academic level, the reactions to a game by an unlikely user are poorly actionable to the game industry. Therefore, it is advisable to select participants from the category of players the game is aimed at, and who would potentially buy the game (McAllister 2012; Tisserand 2010). One way of recruiting motivated players to participate willingly is publishing an advertisement on the publisher or developer website, as has been done by Microsoft Games User Research group, the pioneers in this field, and many others in the following years. Alternatively, it is possible to team up with university research groups that can provide a strong background in methodology, and a very large number of supportive undergraduate gamers.

Since the beginning of user testing activities at Milestone studio in 2008, gamers from outside the development team have been involved by subscribing to the Playtesting Program. The enrolment is free and voluntary, completed by filling in a brief form with personal and contact data, information on gaming platform preference and usage. This information is stored in a database, and referred to at the planning stage in order to find the most appropriate participants for each user session.

Once established that racing videogame players were the target users for *WRC: FIA World Rally Championship* (Black Bean Games 2010), and research questions required a comparison of players with different expertise, two groups of participants were gathered. Rally and off-road games enthusiasts were assigned to the “Expert” group, while subscribers with interests towards other

racing games genres were assigned to the “Novice” group. This selection was carried out among subscribers from the Playtesting Program database, by asking them to complete an additional questionnaire about racing videogames expertise and knowledge.

At this point, the summary table needs a further updates, detailing which groups participants have been considered in the study (Table 16.4).

16.7 Description of the User Testing Sessions

As previously mentioned, game user testing is an iterative process. In order to provide useful and actionable recommendations to the developers, games have to be continuously monitored. Here, the three aspects under examination – realism and approachability of car handling, and stage design – were tested repeatedly over a period of 2 months. Several user testing sessions were scheduled in order to cover all the comparisons listed in the experimental design document and repeated four times, compatibly with the availability of participants and game assets – i.e., some rally stages were not available for testing. The number of participants was set to 5 for each group involved in the tests. In each session, expert and novice rally videogame players were asked to play a number of rally stages “as if they were at home” using standard gaming hardware, and wearing headphones. The presence of driving aids and the stage type (snow, tarmac, gravel, mud) changed depending on the test formulation. Supplementary testing sessions were also set up to collect feedback on menu usability, to compare the game with two competitors, and to test the difficulty of a specific goal-based game mode, the “Rally Academy”.

In these sessions, gameplay activity with background metrics collection was followed by one-on-one discussion with the participants. For a detailed description of a standard playtesting session see Luban (2009a, b) and Fullerton (2008): chapter 9 of the latter includes important information as well as the questions to ask in the discussion of the game experience with participants (p. 253).

16.8 Results

In this paragraph, some of the results that emerged from the analysis of the collected data are presented. In particular, participants’ performances on three special stages are taken into account: one set in Finland, the second from the Jordan rally, and a third stage from the Portugal Rally, set in the famous Estádio Algarve, a football stadium converted into short purpose-built tracks with plenty of hairpins and technical curves.³ The three stages were purposely designed to reflect the road features of

³ This last stage was modified in the retail version of the game, but was included in its sequel, *WRC 2: FIA World Rally Championship* (Black Bean Games 2011).

Table 16.4 The updated table illustrates: (a) the type of data, (b) planned comparisons, (c) which indexes and (d) which participants have been taken into account in this case study in order to test each designers' intents

Designers' intents to test	Realistic car handling	Approachable car handling	Challenging stage design
Subjective data	Interview, questionnaire	Interview, questionnaire	Interview, questionnaire
Objective data	Expected rally maneuvers: drifting, handbrake turning, cutting corners, jumps, high speed Expected better performance by expert rally videogame players in comparison to novices	Expected improvements in completion times, driving accuracy, number and entity of collisions	Expected differences between frequencies of controller activation
Planned comparisons	Expert vs. novice rally videogames players Designers' operationalized intents about rally style racing vs. collected data (when available)	First sector vs. subsequent sectors of different stages Driving aids on vs. driving aids off Expert vs. novice rally videogame players Designers' operationalized intents about expected players' performances vs. collected data (when available)	Fast vs. slow stages Tarmac vs. gravel vs. snow Designers' operationalized intents about expected players' performances vs. collected data (when available)
Indexes to record	Position of the car in respect to the road Speed Drifting index Handbrake activations	Times Driving accuracy index Collision detection Car damage	Frequency of controller activation Position of the car in respect to the road Speed Drifting index Handbrake activations Times Driving accuracy index Collision detection Car damage
Participants	Expert and novice racing videogame players	Expert and novice racing videogame players	Expert and novice racing videogame players

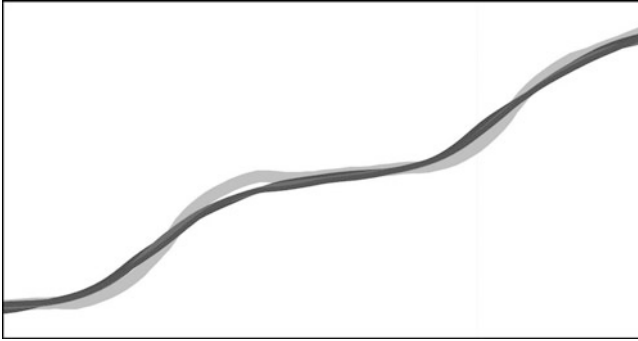


Fig. 16.6 The (mainly overlapping) trajectories of three novice rally videogame players on the fast wide corners of the Finnish special stage. They cut corners and choose a proper driving line. *Dark grey thin lines* refer to car trajectories, the *light grey* refer to the road

the real rallies: the Finnish Rally is famous for its fast gravel roads with numerous straight sections and spectacular jumps between lines of trees and lakeside sceneries; the Jordanian Rally is characterized by a mix of fast and winding dusty roads across the desert and along the coast of the Dead Sea; the Portugal Stage is based on a short, tortuous track within a stadium.

Realistic car handling. The main assumption was that typical rally maneuvers, such as drifting, handbrake turning, corner cutting, and high speed driving would be found throughout the stages. Another issue was investigating whether expert rally videogame players would present more such maneuvers in comparison to novices, since the former are expected to have transferred their pre-existing knowledge of rally games to a new one.

All participants were able to drive in the intended way from their first gameplay session, and many of the expected maneuvers were constantly reported. The amount of drifting was similar between novice and expert drivers across the three different rally stages, as was the occurrence of handbrake turning. Novice participants were also as able as experts to successfully follow effective trajectories, as illustrated in Fig. 16.6.

However, the results of very similar average data patterns between novice and expert participants led to quite different outcomes. Firstly, novice rally players raced at a lower average speed than experts, thus resulting in higher average stage times. This difference is constant across stage sectors and tends to decrease when participants are asked to race on stages of increasing difficulty. The graph in Fig. 16.7 shows how a difference of nearly 20 km/h in the Finnish stage is reduced to 12 km/h in the winding Jordan stage, and further to a mere 6 km/h in the tortuous Portuguese stage. Thus, the average speed values appropriately reflected the layout for each stage and their growing challenge.

Although expected, the difference in speed mirroring players' expertise was somehow puzzling, since expert and novice participants managed to drift the same amount. The problem was solved with the examination of a second index, the driving accuracy index: it is not a matter of how much drifting occurred, but how it is

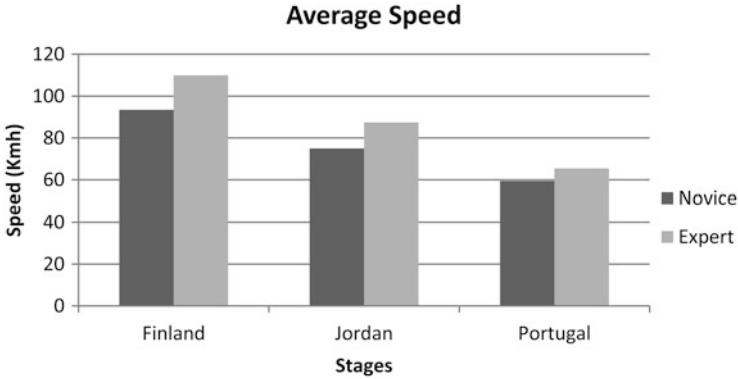


Fig. 16.7 Average speed relative to novice and expert rally videogame players across three different stages: the former drove constantly slower than the latter. The differences between the three rally stages mirror their layout complexity: the Finland stage is full of long straight sections and wide fast corners, the Jordan stage is characterized by sandy gravel terrain and snake curves within canyons, while the Portuguese stage is a tortuous track within a football stadium

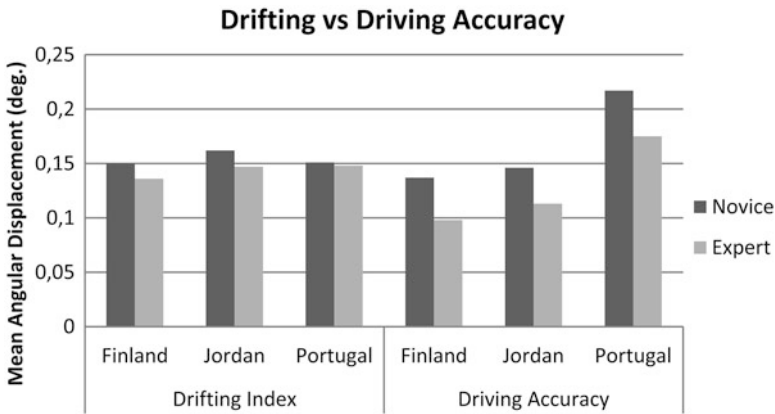


Fig. 16.8 One index is never enough. While novice and expert participants drifted to a similar extent, the outcome in driving accuracy is constantly very different: *shorter bars* correspond to a better performance – smaller mean angular displacement from the *ideal line*, thus to higher accuracy

employed. The graph in Fig. 16.8 shows that the performance of experts was constantly more accurate in comparison to novices, despite the similar amount of drifting. Indeed, the mean angular displacement from the ideal trajectory is lower for expert participants while the amount of drifting is the same.

Further analysis performed on brake usage data showed also a recurring pattern: while experts brake earlier when entering corners, and drive through them drifting at high speed with the car sideways, novices brake later in the corners thus drifting

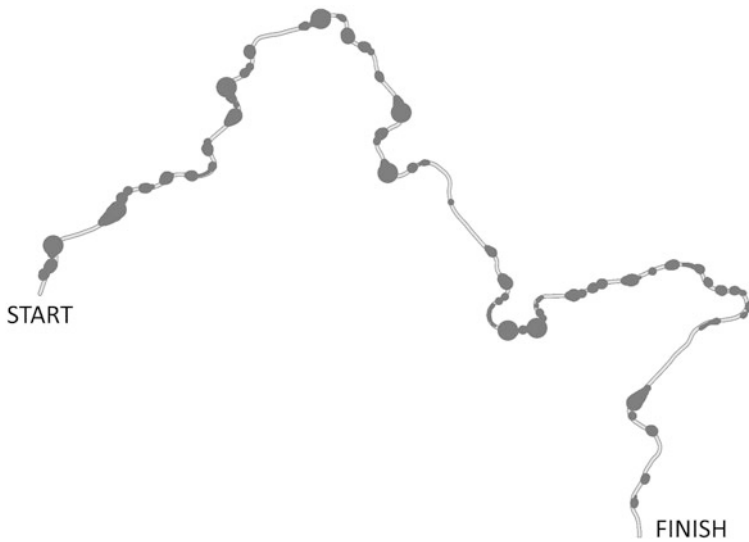


Fig. 16.9 A braking heat map illustrates where a player braked more frequently and more heavily while racing the Jordan special stage. The first information is portrayed by the position of the *dark grey spots*, while the latter by their size. The *light grey* tortuous line represent the road

while exiting corners at a much slower speed. This important result led to two crucial conclusions. Since this different way of tackling corners between novice and expert drivers is also a major issue in real motorsports (Bentley 1998), this result contributed to raise confidence in the realism of the handling model, and consequently a realistic gameplay experience. On the other hand, this was also an important indication on how to develop driving aids capable of providing novice players with an authentic and satisfying driving experience.

This result has been achieved by plotting players' controller activations directly on the maps in the game, according to the "heat maps" technique employed in many videogames research studies, and already presented in this volume. The graph in Fig. 16.9 illustrates how the activation of brakes has been superimposed on the map of the Jordan special stage in order to find critical points. It is indeed possible to consider several different events on the maps, such as collisions. Typical examples of the application of this technique are the "Death Maps", where locations of players' deaths are overlapped to the maps of each game level, to show where players died most often (cf. Valve 2003). This is the first published example of heat maps applied to a racing videogame.

In order to correct the late braking issue by inexperienced players, designers decided to intervene in two distinct ways. First, pacenotes indications were anticipated in the easiest game settings: the idea was that by notifying players earlier of upcoming sharp corners would give them more time to react and brake at the correct time. However, since further indications from subjective feedback suggested that novice players were less inclined to rely on pacenotes than expert players, a second

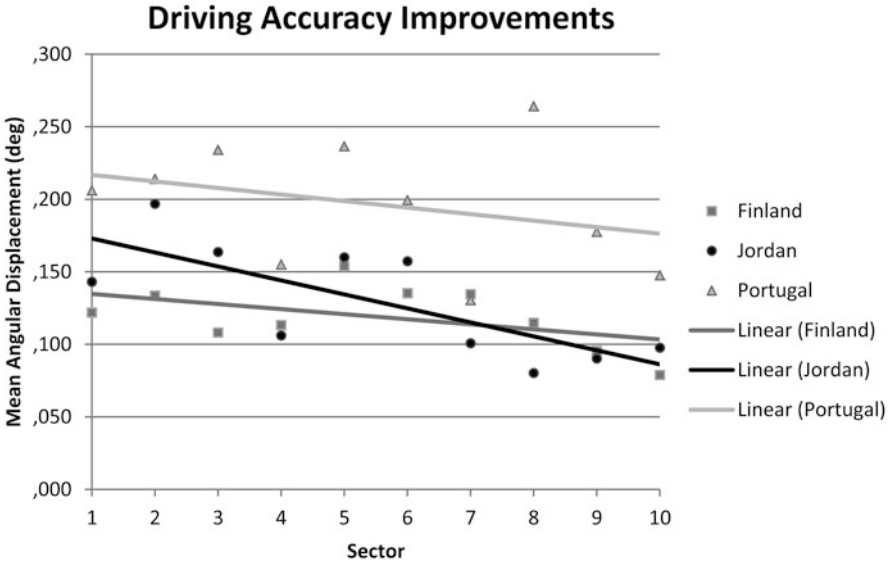


Fig. 16.10 Driving accuracy scores across different sections of three different rally stages, collected from a group of novice rally gamers in their first run

modification was made to the driving help. Instead of implementing a braking aid that greatly enhances the power of the brakes, designers introduced an “auto-braking” system, which makes the car automatically slow down before corners. Only the latter solution allows novice players to experience the correct braking points and the resulting way of driving through corners.

Approachable Car Handling. In order to assess whether car handling was offering a friendly approach to players, a comparison between their performances throughout each stage was carried out. The data from each stage were split into ten equal-sized sections. Since the division was based on percentage of stage completion, it was possible to compare stages with different layout or length.

Figure 16.10 presents changes in driving accuracy by a group of novice participants in their first run across the three different stages. Each dot represents the average score relative to driving accuracy for each section of each stage. The higher the score, the greater is the displacement from the optimal driving line. The three lines sum up the trend of each set of data: all of them are tilted to the right, thus indicating that the more the participants play, the smaller the displacement is. In other words, their performances are increasingly accurate. The Jordanian stage presents the biggest improvement: the line is tilted more than the other two.

Challenging Stages Design. Several game metrics, and in particular the activations and modulations of gas, brakes, steering wheel and handbrake, reflect how participants have interpreted the overall configuration of each stage. Stages with many

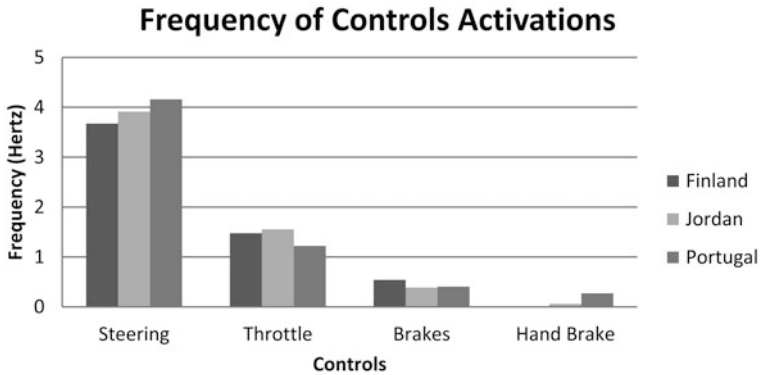


Fig. 16.11 The average frequency of control activations in three different rally stages

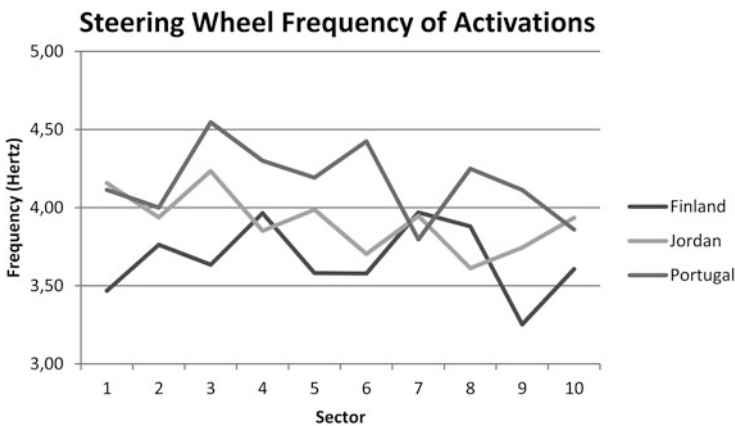


Fig. 16.12 The average frequency of steering wheel activations across three different rally stages

long straight sections and fast wide turns require less acting on the control device in comparison to stages with many bends and hairpin turns. Comparing these indexes should give a raw indication about their diversity. Finding no differences would indicate that driving across supposedly different stages actually leads to a very similar driving experience.

Significant differences were found both by comparing the three stages, and by taking into account the different sectors of each stage. For example, the Finnish stage required a smaller amount of steering adjustments than the Jordanian stage, and both required fewer adjustments than the Portuguese one (Fig. 16.11). Moreover, by examining how mean frequency values change across the different sectors, it is possible to make further observations: for example, the Finnish stage began with a less demanding first sector in comparison to the other two stages, while the Jordanian stage seems to be the more consistent between the three (Fig. 16.12). The findings relative to each sector have been of particular interest

during the design of the rally school game mode. In that case, designers had to build a series of challenges of growing difficulty where players are asked to perform specific tasks (e.g., follow a given trajectory, use the handbrake at specific points, race through a sector under a certain time, etc.) with the purpose of getting acquainted with the car handling. It is important to choose wisely where to set these challenges in order to offer an appropriate level of difficulty to players that are not yet familiar to the gameplay.

As previously mentioned, a second hypothesis about frequency of control activations regards the comparison between novice and expert rally videogame players. The first are expected to have more difficulties in controlling the car, thus presenting a higher frequency of control activation, due to continuous adjustments, in comparison to the latter. However, the data collected painted a different picture: novices modulated the controls less frequently than experts. In particular, they consistently applied fewer adjustments to the steering wheel and the gas pedal in the three stages. A (second) puzzling and unexpected result at first, this index actually reflects an important difference between driving on dirt and driving on tarmac. In the latter case, since the tires have greater grip, the driver has to be very precise in braking, setting a trajectory and opening the gas at the right moment while exiting corners; even more so on straight sections, where small steering corrections may cause the vehicle to skid and consequently lose control. Driving on dirt surfaces implies much less grip between the tires and the road surface: the car is very “floaty” and even on straight sections continuous left-right steering adjustments are required in order to keep it straight at high speeds. The fact that novice players acted less frequently on the controls made the developers rethink the design of a second driving aid: the stability assistant. On asphalt, this help filters the controller input in order to exclude smaller involuntary modifications, with the consequence of improving vehicle stability and thus control. On dirt, the stability help had to operate differently, taking into account the necessary trajectory adjustments.

16.8.1 Average or Raw Data?

In the results paragraph, mean values were frequently presented. While computing and comparing mean values are necessary when examining similarities and differences between groups of different participants (e.g., novice vs. expert players), or between different instances (e.g., the Finnish vs. Jordanian stage), it is recommendable to also look through raw data. As already mentioned, mean values tend to hide unique potential issues, even though these may not be representative of the group. Figure 16.13 gives such an example. The trajectories indicate how three expert participants encountered problems at the same point in the Jordanian stage. Since the rest of their driving was accurate and very fast, these events were likely to pass unnoticed by examining mean values. The examination of single trajectories with the support of video recordings made it possible to discover that these three players had crashed into barely visible small rocks at both sides of this particular stretch of

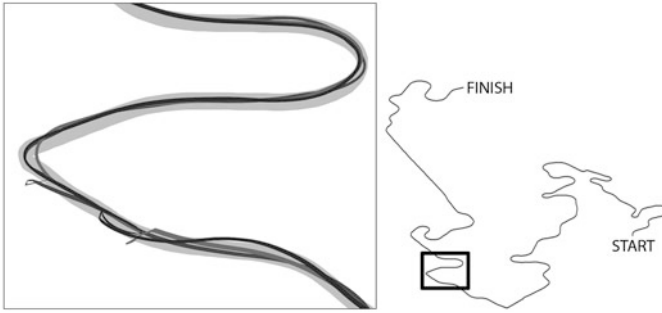


Fig. 16.13 Three different expert participants crashed in the very same stretch of road: something is definitely wrong in the stage design. *Left*: the three trajectories (*dark grey lines*) show high speed impacts with road side objects (the thickness of the lines reflects the speed of each vehicle). *Right*: the frame identifies the location of the crashes on the stage map

road. This issue was reported to designers for further consideration, suggesting a change in track design – removing the rocks, or at least making them more visible – or making their presence evident through pacenotes.

16.9 Metrics After the Game Has Shipped

The gameplay metrics discussed until now belong to the development phase of the game, when it is still possible to apply substantial changes to the game code. What happens when the game has shipped? It is indeed possible to take advantage of game metrics to investigate how buyers of the game actually use it in their homes (for a review, see Medler and Magerko 2011). Telemetry data, or metrics measured at a distance, are widely employed to monitor single and multiplayer game usage (Phillips 2010; Medler et al. 2011; Hullett 2012), as well as to directly test with users specific sections of the game – in *Left 4 Dead* (Valve Corporation 2008), levels of the game are frequently tested this way. The examination of retail game usage metrics is the last option to recognize problems and consider a fix through a game update, but it is also an important source of information for improving the design of following titles. In an interesting paper, Phillips (2009) acknowledged the relatively high rate of game quitting and discussed the need of design changes to support and motivate playing.

Game telemetry systems are usually provided by game publishers, but developers and GUR researchers can indeed participate in their implementation. Possibly the easiest ways of collecting this type of data is by using reward systems such as Xbox Live Achievements, PlayStation Trophies, or Steam Achievements. These have proved crucial for the success of a game, and are now a consolidated standard (McClanahan 2009; Blair 2011; Jakobsson 2010, 2011). One of the most interesting

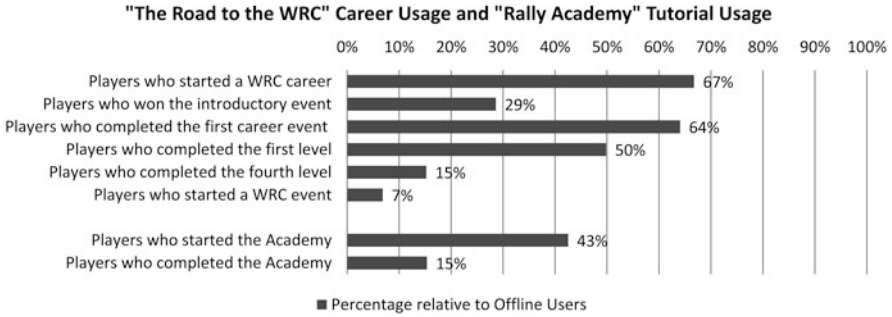


Fig. 16.14 Description of game usage by actual users shortly after the release of the game. The numbers represent the percentage of total users that completed the corresponding task

characteristics of these reward systems is that videogame developers have complete freedom in their design. They can decide what players have to do, how many times, under which circumstances, etc., in order to obtain the reward. Designers, therefore, aim to create imaginative and clever achievements in order to inspire and entertain players with original and uncommon challenges. However, game researchers can team up with them and take advantage of this freedom to monitor players' behaviors and preferences: in fact, with an appropriate design it is possible to monitor game usage such as when players used a specific game mode or a game feature for the first time, how much game modes and features are used, and whether players have managed to complete a certain event. This has been done for WRC, and Fig. 16.14 displays the data relative to the usage of the career mode shortly after the release date on the Xbox360 console. The career consists of an introductory rally event, followed by several other levels consisting in a number of different races the player has to participate in order to earn money and prestige to climb from the entry level class of cars up to the WRC. One data worth commenting upon is the very low percentage of players who won the introductory event: this suggests an excessive difficulty of this particular event. However, when considering the even lower percentage of players who went through the tutorial mode (which was skippable at the beginning of the career), it is indeed possible to suggest designers to lower its difficulty or make the tutorial compulsory. Once again it is proven that players want to directly jump in the game.

16.10 Conclusions

By employing game metrics during the user testing of *WRC: FIA World Rally Championship* (Black Bean Games 2010), we were allowed to verify the designers' intents while, at the same time, we built a deep and detailed understanding of how

players interacted with the videogame when it was still under development. Game telemetry metrics provided a further step by recording and assessing actual users' game usage. In the first case, game metrics functioned as a tool to quickly collect objective feedback with the highest granularity possible, while in the second case telemetry metrics were disguised as game achievements in order to monitor some aspects of actual game usage. We found many interesting results by integrating objective data from metrics with opinions collected in interviews, but more importantly several issues were identified that would have been impossible to detect through subjective feedback analysis alone. These findings were put directly to good use and contributed to the improvement and refinement of the videogame: the car handling model was refined with progressive tuning, driving aids were re-designed to better meet the needs of inexperienced rally gamers, thus supporting game approachability, learning curves were incorporated into the rally stages, and finally some suggestions were put forward to improve stage design. Every finding also contributed to the design of the next iteration of the game, while the technology developed for collecting metrics has been refined and successively applied to the following titles by Milestone studio. For example, when *WRC 2: FIA World Rally Championship* (Black Bean Games 2011) playtesting began 1 year later, the telemetry system was employed again with small changes, mostly due to the new features included in the game.

But in the end, what impact did the employment of game metrics have on the game and its reception by gamers? It is possible to answer this question by examining game reviews, an important source of information for gamers for understanding a game, its gameplay and technical features, how it compares with other games, etc. (Zagal et al. 2009). It is also a well-known fact that videogame reviews have a strong effect on potential buyers. While it does not directly affect play experience, reading reviews before playing has a strong influence on gamers' post-play evaluations of the game quality (Livingston et al. 2011a, b). In other words, reading a negative review before playing may bias the evaluation of an otherwise good game experience (Raita and Oulasvirta 2011), just as much as when considering purchasing the game. The car handling model – the most important aspect of a driving game and the main concern of the user tests covered in this chapter – has been judged by international reviewers as the best part of the game: “its driving reveals itself to be entertaining and engaging” (Robinson 2010), “featuring a deep and rewarding handling model” (Barron 2010). Approachability was also welcomed (“it also manages to remain accessible while retaining the excitement that simulation fans expect” – *ibid*), as well as the “competent” and “challenging” stage design (Sanchez 2010). In sum, despite some flaws in the technical department, the core gameplay was favorably received with review scores up to 80 out of 100, and part of this success can be credited to user-testing with game metrics.

In the end, the employment of metrics has been a powerful step forward in game user research at Milestone studio, thanks to the amazing level of detail it provides and in particular to the way it speeds up data analysis and understanding, which is great for a medium sized studio working on three projects a year.

16.11 Takeaways

- Plan in advance the design of the user tests, the comparisons and the statistics needed to answer the research questions. This will save you time later.
- Ask yourself “Why I am recording this index?” for every index you are considering. If you can’t answer within 3 s, the index will be scarcely useful.
- A single source of data is not enough: always collect subjective feedback in addition to metrics.
- Create new indexes when the game does not provide the right ones. Sometimes the parameters within the game are not sufficient to track player behavior in the best way. Here, the “driving accuracy index” was included for user testing purposes only (although it may function perfectly as an in-game feature).
- The direct involvement of developers in the process of designing the metrics collector will provide useful information and promote their interest and collaboration.
- Be prepared for issues and last-minute changes: game development is a very complex activity, and you will need some flexibility to cope with levels that won’t load or are not ready for testing – but your metrics collector may fail as well. It is better to collect partial feedback than no feedback at all.
- Test everything before the actual user tests start; if everything works, do not make modifications until the test ends.
- Always compare results between multiple indexes.
- Metrics are fast to analyze, giving you more time to process the results.
- Look at the raw data whenever possible.
- Video recordings are a good support for metrics and a great backup plan if something goes wrong.
- Developers are not analysts, so be sure your results are clearly explained with graphics and tables. Developers are developers, so be sure to present useful and actionable results.

About the Authors

Pietro Guardini works as games user researcher at Milestone, the major game development studio in Italy, where he started and is in charge of user testing activity. He began working on Human Computer Interaction as an undergraduate student in 1999 in the Virtual Reality Lab of the Department of Experimental Psychology at Padova University (Italy). After obtaining a Ph.D. in Experimental Psychology in 2008 from the same university, he started his collaboration with Milestone working on many titles, including: *MotoGP 08*, the *Superbike World Championship* (SBK) and the *World Rally Championship* (WRC) series of videogames. He also is lecturer in Research Methodology in Human Factors at the Bicocca University in Milan, Italy.

Paolo Maninetti worked as senior game programmer at Milestone, the most important game development studio in Italy, where he was in charge of A.I. Programming. He worked on titles such as *MotoGP08*, the *Superbike World Championship* (SBK) and the *Superstars V8* series of videogames. He has also contributed to *WRC: FIA World Rally Championship 2010* game as Gameplay Programmer, and developed the game metrics data logger mentioned in this chapter. In the past Paolo worked in Ubisoft Milan where he took part in the production of games such as *Rayman M*, *Tomb Raider: The Prophecy* and *Beyond Good and Evil*. He holds a M.A. in Computer Science from the University of Milan, Italy. Paolo is currently working as Game Programmer at PopCap Games on the *Bejeweled* franchise.

Acknowledgments The authors would like to thank Magy, Anders and Alessandro – the editors of this volume – for the great work and their effort, constant availability and understanding. The authors are also grateful to the anonymous reviewers of this chapter for their valuable suggestions, which resulted in a stronger and improved chapter. A big “Thank you!” to our great colleagues at Milestone. Pietro would like to thank the Games User Research (GUR) community for providing constant inspiration.

References

- Accot, J., & Zhai, S. (1997). Beyond Fitts' law: Models for trajectory-based HCI tasks. In *Proceedings of CHI97*, Atlanta, GA, USA.
- Accot, J., & Zhai, S. (1999). Performance evaluation of input devices in trajectory-based tasks: An application of the steering law. In *Proceedings of CHI '99*, Pittsburgh, PA, USA.
- Amaya, G., Davis, J. P., Gunn, D., Harrison, C., Pagulayan, R. J., Phillips, B., & Wixon, D. (2008). Games User Research (GUR): Our experience with and evolution of four methods. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advice from the experts for advancing the player experience*. San Francisco, CA, USA: Morgan Kaufmann.
- Ambinder, M. (2009). Valve's approach to playtesting: The application of empiricism. In *Proceedings of game developer's conference 2009*. San Jose, CA, USA.
- Ambinder, M. (2011). Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. In *Proceedings of game developer's conference 2011*. San Jose, CA, USA.
- Barron, J. (2010). *WRC FIA World Rally Championship Review*. Review published on Game Spot UK. Retrieved here: <http://www.gamespot.com/wrc-fia-world-rally-championship/reviews/wrc-fia-world-rally-championship-review-6281451/>
- Bentley, R. (1998). *Speed secrets: Professional race driving techniques*. Osceola, WI, USA: MBI Pub. Co.
- Bernhaupt, R. (2010). *Evaluating user experience in games, human interaction series*. London, UK: Springer.
- Bethke, E. (2003). *Game development and production*. Plano, TX, USA: Wordware Publishing.
- Blair, L. (2011). The cake is not a lie. How to design effective achievements. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/6360/the_cake_is_not_a_lie_how_to_php
- Blythe, M. A., Overbeeke, K., Monk, A. F., & Wright P. C. (Eds.) (2003). *Funology: From Usability to Enjoyment*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Bogost, I. (2011). *How to do things with videogames*. Minneapolis, MN, USA: University of Minnesota Press.
- Bojko, A., Buttimer, J., & Zace, S. (2010). Preparation. In R. M. Schumacher (Ed.), *Handbook of global user research*. Burlington, MA, USA: Morgan Kaufmann.

- Clark, N. (2010). Psychology is fun. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/6145/psychology_is_fun.php
- Collins, J. (1997). Conducting in-house playtesting. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/3211/conducting_inhouse_play_testing.php
- Davis, J. P., Steury, K., & Pagulayan, R. (2005). A survey method for assessing perceptions of a game: The consumer playtest in game design. *Game Studies*, 5(1). Retrieved here: http://www.gamestudies.org/0501/davis_steury_pagulayan/
- Excel (Computer Software). (2010). Redmond, WA, USA: Microsoft Corporation.
- Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 381–391.
- Forza Motorsport 2 (Computer Software). (2007). Redmond, WA, USA: Microsoft Games Studios.
- Fullerton, T. (2008). *Game design workshop: A playcentric approach to creating innovative games* (2nd ed.). Burlington, MA, USA: Morgan Kaufmann.
- Fulton, B. (2002). Beyond psychological theory: Getting data that improves games. In *Proceedings of Game Developers Conference GDC 2002*. San Jose, CA, USA. Retrieved here: http://www.gamasutra.com/gdc2002/features/fulton/fulton_01.htm
- Fulton, B., & Romero, R. (2004). User-testing in a hostile environment: Overcoming resistance and Apathy in your game company. In *Proceedings of Game Developers Conference GDC 2004*. San Jose, CA, USA.
- Fulton, B., & Medlock, M. (2003). Beyond focus groups: Getting more useful feedback from consumers. In *Proceedings of Game Developers Conference GDC 2003*. San Jose, CA, USA.
- Greenwood-Ericksen, A., Preisz, E., & Stafford, S. (2010). Usability breakthroughs: Four techniques to improve your game. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/6130/usability_breakthroughs_four_.php
- Griffiths, G. (2009). Practical game playtesting: A Wii-based case study. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/3894/practical_game_playtesting_a_.php
- Guardini, P. (2002). Virtual unreality of videogames. *Psychology Journal*, 1(1), 57–70.
- Halo 2 (Computer Software). (2004). Redmond, WA, USA: Microsoft Games Studios.
- Hecker, C. (2000). Physics in computer games. *Communications of the ACM*, 43(7), 35–39.
- Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events. *ACM Computing Surveys (CSUR)*, 32(4), 384–421.
- Hopson, J. (2001). Behavioral game design. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/3085/behavioral_game_design.php
- Hullett, K., Nagappan, N., Schuh, E. & Hopson, J. (2012). Empirical analysis of user data in game software development. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Lund, Sweden.
- Ip, B., & Jacobs, G. (2004). Quantifying game design. *Design Studies*, 25(6), 607–624.
- Isbister, K. (2006). *Better game character by design: A psychological approach*. San Francisco, CA, USA: Morgan Kaufmann.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advice from the experts for advancing the player experience*. San Francisco, CA, USA: Morgan Kaufmann.
- Isokoski, P., & Martin, B. (2007). Performance of input devices in FPS target acquisition. In *Proceedings of ACE 2007*. New York City, NY, USA.
- Jakobsson, M. (2010). Achievement design: Lessons from an Xbox live community study. In *Proceedings of GDCE Game Developers Conference Europe 2010*. Cologne, Germany.
- Jakobsson, M. (2011). The achievement machine: Understanding Xbox 360 achievements in gaming practices. *Game Studies*, 11(1). Retrieved here: <http://gamestudies.org/1101/articles/jakobsson>
- Kuniavsky, M. (2003). *Observing The User Experience – A Practitioner’s Guide to User Research*. San Francisco, CA, USA: Morgan Kaufmann.
- Laitinen, S. (2005). Better games through usability evaluation and testing. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/2333/better_games_through_usability_.php?page=1

- Left 4 Dead (Computer Software). (2008). Bellevue, WA, USA: Valve Corporation.
- Livingston, I. J., Nacke, L. E., & Mandryk, R. L. (2011a). *The impact of negative game reviews and user comments on player experience*. ACM SIGGRAPH 2011 game papers. Vancouver, BC, Canada.
- Livingston, I. J., Nacke, L. E., & Mandryk, R. L. (2011b). Influencing experience: The effects of reading game reviews on player experience. In *10th International Conference on Entertainment Computing (ICEC 2011)*. Vancouver, BC, Canada.
- Looser, J., Cockburn, A., & Savage, J. (2005). On the validity of using first-person shooters for Fitts' law studies. In *Studies, People and Computers XIX (Volume 2): British computer society conference on human computer interaction*, Edinburgh, Scotland.
- Luban, P. (2009a). The silent revolution of playtests, Part 1. *Gamasutra Feature*. Retrieved at: http://www.gamasutra.com/view/feature/3963/the_silent_revolution_of_.php
- Luban, P. (2009b). The silent revolution of playtests, Part 2. *Gamasutra Feature*. Retrieved at: http://www.gamasutra.com/view/feature/3985/the_silent_revolution_of_.php
- McAllister, G. (2012). Wrong is often right. *Column in Edge Online*. Retrieved here: <http://www.edge-online.com/opinion/wrong-often-right>
- McAllister, G., & White, G. R. (2010). Video game development and user experience. In R. Bernhaupt (Ed.), *Evaluating user experience in games, human interaction series*. London, UK: Springer.
- McClanahan, G. (2009). Achievement design 101. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/blogs/GregMcClanahan/20091202/3709/Achievement_Design_101.php
- Medler, B., & Magerko, B. (2011). Analytics of play: Using information visualization and game-play practices for visualizing video game data. *Parsons Journal of Information Mapping*, 3(1). Retrieved here: http://pjim.newschool.edu/issues/2011/01/pdfs/ParsonsJournalForInformationMapping_Medler-Ben+Magerko-Brian.pdf
- Medler, B., John, M., & Lane, J. (2011). Data cracker: Developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of CHI '11*, New York City, NY, USA: ACM.
- Natapov, D., Castellucci, S. J., & MacKenzie, I. S. (2009, May 25–27). ISO 9241-9 evaluation of video game controllers. In *Proceedings of the graphics interface conference*, Kelowna, British Columbia, Canada.
- Pagulayan, R. J., Steury, K. R., Fulton, B., & Romero, R. L. (2003). Designing for fun: User-testing case studies. In M. Blythe, K. Overbeeke, A. Monk, & P. Wright (Eds.), *Funology: From usability to enjoyment* (pp. 137–150). Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Parker, L. (2011). The science of playtesting. *Gamespot Feature*. Retrieved here: <http://www.gamespot.com/features/the-science-of-playtesting-6323661/>
- Phillips, B. (2009). Staying power: Rethinking feedback to keep players in the game. *Game Developer Magazine*, 16(6). Retrieved Online at Gamasutra.com: http://www.gamasutra.com/view/feature/4171/staying_power_rethinking_feedback_.php?print=1
- Phillips, B. (2010). Peering into the black box of player behavior: The player experience panel at Microsoft game studios. In *Proceedings of game developer's conference 2010*. San Jose, CA, USA.
- Picard, R. W. (1997). *Affective computing*. Cambridge, MA, USA: MIT Press.
- Pure (Computer Software). (2008). Glendale, CA, USA: Disney Interactive Studios.
- Raita, E., & Oulasvirta, A. (2011). Too good to be bad: Favorable product expectations boost subjective usability ratings. *Interacting with Computers*, 23, 363–371.
- Robinson, M. (2010). *WRC FIA World Rally Championship Review: Kickin' up the dirt*. Review published on IGN UK. Retrieved here: <http://uk.ps3.ign.com/articles/112/1127695p1.html>
- Romero, R. (2008). Successful instrumentation: Tracking attitudes and behaviors to improve games. In *Proceedings of Game Developers Conference GDC 2008*. San Jose, CA, USA.
- Sanches, J. D. (2008). *The driving games manual: The ultimate guide to all car-based computer and video games*. Yeovil, Somerset, UK: Haynes Publishing.
- Sanches, J. D. (2010). *WRC FIA World Rally Championship Review*. Retrieved here: <http://www.eurogamer.net/articles/2010-10-08-wrc-fia-world-rally-championship-review>
- SBK2001 (Computer Software). (2000) Redwood City, CA, USA: EA Sports.

- Schell, J. (2008). *The art of game design: A book of lenses*. Burlington, MA, USA: Morgan Kaufmann.
- Schuh, E., Gunn, D. V., Phillips, B., Pagulayan, R. J., Kim, J. H., & Wixon, D. (2008). TRUE Instrumentation: Tracking real-time user experience in games. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advice from the experts for advancing the player experience*. San Francisco, CA, USA: Morgan Kaufmann.
- Schumacher, R. M. (2010). *Handbook of global user research*. Burlington, MA, USA: Morgan Kaufmann.
- Snitker, T. V., & Jeffers, J. (2010). User research throughout the world. In R. M. Schumacher (Ed.), *Handbook of global user research*. Burlington, MA, USA: Morgan Kaufmann.
- Speyrer, D., & Jacobson, B. (2006). Valve's design process for creating half-life 2. In *Proceedings of game developers conference*. San Jose, CA, USA.
- SPSS (Computer Software). (2011) New York City, NY, USA: International Business Machines Corporation.
- Tableau (Computer Software). (2003) Seattle, WA, USA: Tableau Software.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired*, 15(9). Retrieved here: http://www.wired.com/gaming/virtualworlds/magazine/15-09/ff_halo?currentPage=all
- Tisserand, D. (2010). PlayStation: Evolving user testing to social, casual and portable gaming. In *Proceedings of Game Developers Conference GDC 2010*. San Francisco, CA, USA.
- Tullis, T., & Albert, B. (2008). *Measuring the user experience: Collecting, analyzing, and presenting usability metrics*. Burlington, MA, USA: Morgan Kaufmann.
- Valve Corporation, (Computer Software). (2003). *Steam statistics*. Retrieved here: <http://store.steampowered.com/stats/>
- Van der Heijden, J. (2010). Successful playtesting in swords & soldiers. *Gamasutra Feature*. Retrieved here: http://www.gamasutra.com/view/feature/5939/successful_playtesting_in_swords_.php
- WRC 2: FIA World Racing Championship (Computer Software). (2011). Varese, Italy: Black Bean Games.
- WRC: FIA World Racing Championship (Computer Software). (2010). Varese, Italy: Black Bean Games.
- Zagal, J. P., Ladd, A., & Johnson, T. (2009). Characterizing and understanding game reviews. In *Proceedings of the 4th international conference on foundations of digital games – FDG '09*, New York City, NY, USA: ACM Press.
- Zammito, V. (2011). The science of play testing: EA's methods for user research. In *Proceedings of game developer's conference 2011*. San Francisco, CA, USA.

Part IV

Metrics Visualization

This part of the book deals with visualization methods of game metrics as a way of analyzing and reporting the data to stakeholders. Bioware, Electronic Arts and Pixel Ante share the solutions that were devised for games such as *Dead Space 2*, *Dragon Age: Origins* and *Pixel Legions*. This panoramic view of different visualizations systems covers genres as different as:

- Third person action adventures
- Real time strategy games
- Role playing games

This part has the following take-aways:

- Provide an introduction to the areas of visualization and visual analytics
- Provide case studies on the use of visualization and visual analytics as methods for game data analysis

The part will consist of four chapters:

- Chapter 17: *Spatial Game Analytics* introduces the area of spatial and temporal analysis which is important for game telemetry data as the data is often spatial and temporal. The chapter is a contribution from Anders Drachen (co-editor of this book) with Matthias Shubert who is a professor at Ludwig-Maximilians-Universität.
- Chapter 18: *Visual Game Analytics* discusses the use of visual analytics for games and provides case studies of work done at Electronic Arts. The contribution is from Ben Medler, Ph.D. student at Georgia Tech University.
- Chapter 19: *Visual Analytics Tools – A Lens into Player’s Temporal Progression and Behavior* provides a discussion of new visual analytics tools developed to service designers at Electronic Arts and Pixel Ante. The chapter is a contribution from Magy Seif El-Nasr (co-editor of this book) and Andre Gagné, user researcher at THQ, Dinara Moura, Ph.D. student at Simon Fraser University, Bardia Aghabeigi, Ph.D. student at Northeastern University and game analytics researcher at Blackbird Interactive.
- Chapter 20: *Interview with Nicklas “Niffles” Nygren*. Nicklas Nygren is an archetypal independent game developer working in Sweden and Denmark. The interview introduces his views on game analytics as an independent developer.

Chapter 17

Spatial Game Analytics

Anders Drachen and Matthias Schubert

Take Away Points:

1. Introduction to spatial game analytics and the current state-of-the-art in games development and games research.
2. Overview of spatial analytics outside games and recommendations for the application of new methods in game design and –development.
3. Advice and ideas on how to get started with spatial analysis of behavioral game telemetry data, which methods to use when, fundamental considerations and a large number of examples from several different game genres to draw inspiration from.

17.1 Introduction

Perhaps the most beloved visualization of player behavior is the heatmap, which offers clear and intuitive feedback about the spatial behavior of players. Heatmaps are, however, only the tip of a very deep iceberg of the area – we here will refer to as **spatial game analytics** – and it has a lot more to offer than heatmaps, not the least a strong explanatory power for deciphering and understanding player behavior. Here we take a plunge into these deep waters, exploring what is already being done

A. Drachen, Ph.D. (✉)
PLAIT Lab, Northeastern University, Boston, MA, USA

Department of Communication and Psychology, Aalborg University, Aalborg, Denmark
Game Analytics, Copenhagen, Denmark
e-mail: andersdrachen@gmail.com

M. Schubert, Ph.D.
Institute for Informatics, Ludwig-Maximilians-Universität, Munich, Germany

and what can be done within this area and to a lesser degree visualization – which is further explored in the following chapters (Chaps. 18 and 19).

When using the term “spatial analytics” in the context of computer games, what is usually meant is evaluation or analysis of the spatial component of player behavior obtained via telemetry data (Drachen and Canossa 2011). This is applicable to all games, as it is difficult to point to a computer game that does not integrate some form of spatial behavior or mechanics, from simple vector-based navigation over 2D environments in a side-scroller to the fully-fledged 3D environments of MMORPGs and online action/shooters. The spatial component of such games forms the basis for the experience of the player, e.g. navigating in an environment. Therefore, the analysis and evaluation of the spatial behavior of the player is of direct interest to game design and Game User Research (GUR) (Isbister and Schaffer 2008) [for an introduction see: Lewis-Ewans (2012)].

Consider for example situations where we want to evaluate if: (a) a game design works; (b) if the play experience matches the intent; (c) if players behave as expected or (d) how the spatial layout of the game and its component can be optimized to improve team-based activities or conversion rates. In all of these situations, taking into account the spatial aspects of player behavior adds layers of depth and context not possible using only non-spatial information. For example, knowing that players took longer than expected to complete a game level does not tell us why this happened, but spatial analytics can be used to factor causes of the delay, and pinpoint where players are experiencing problems progressing, e.g. Thompson (2007), Ramero (2008), Miller and Crowcroft (2009), Drachen and Canossa (2011), Zoeller (2010, 2011), and Dankoff (2012).

Spatial analysis – i.e. the analysis of any data containing a geospatial component where this component is a part of the analysis – often results in visualizations that are intuitive to understand and work with for the variety of stakeholders in a game development company or game publisher. The result of spatial analytics can take many forms, for example a simple heatmap. A heatmap provides a readily interpretable overview of player behavior, because it puts the behavioral data into a concrete game context, providing instantly actionable insights (Kim et al. 2008; Drachen and Canossa 2011). Figure 17.1 shows an example of this, where the density of locations where players have requested an extraction in the sandbox-style shooter *Just Cause 2* (Square Enix, 2010). Using this kind of visualization, it is possible to evaluate whether the pattern of behavior matches that intended by the games’ design. Another example: evaluating why players follow a particular strategy in an instanced raid zone in a 3D MMORPG is possible by looking only at the numbers. However, when those numbers are mapped directly on top of or directly into the game environment, and correlations between the variables shown (e.g. where raid members position themselves during boss fights), a perhaps more intuitively understandable way of interacting with the data is gained. This in turn leads to a better understanding of how to prevent or promote specific behaviors (see also Chap. 19).

Another component of player-derived behavior telemetry data from computer games that often ties in directly with spatial behavior is the **temporal dimension** of the data (also discussed in depth in Chap. 19). Spatial behavior – indeed all game

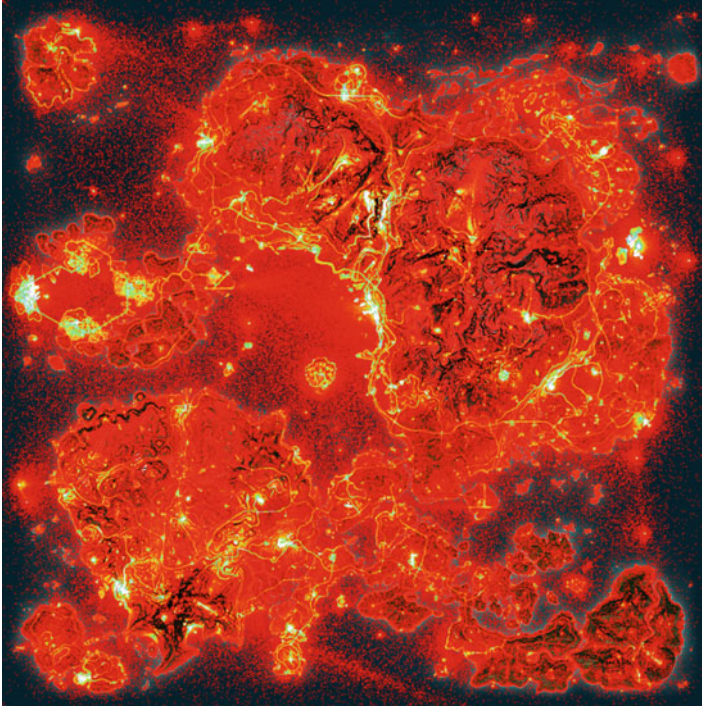


Fig. 17.1 An example of a spatial visualization of behavior data: A heatmap of extraction events in *Just Cause 2*. More than 22.3 million events form the basis for the heatmap. Color ramp is scaled to the cell with the largest value is always *white* (© Square Enix Europe; courtesy of Square Enix Europe. The heatmap is developed by Jim Blackhurst)

play – occurs over time, and it is therefore essential to integrate the temporal dimension in game analytics. This is currently also the case, with many standard Key Performance Indicators (KPIs) from social online games being aggregated as a function of time, e.g. Daily Active Users (DAU) or Monthly Active Users (MAU) (see Chap. 4). Similarly, the temporal aspects of player behavior is vital to any kind of progression analysis (Drachen et al. 2009), path analysis (Miller et al. 2009), bot detection analysis (Thureau et al. 2003; Thawonmas et al. 2008), time spent analysis DeRosa (2007), navigation analysis (Moura et al. 2011) and strategy analysis (Gagné 2011; Gagné et al. 2012) and many other forms of analysis based on player behavior in the industry or academia (see Chaps. 18 and 19). When combining spatial and temporal information with the telemetry data obtained about player behavior (e.g. asset use, events, etc.), a combination of measures is gained, which when analyzed jointly provide powerful insights into the behavior of players and a game’s design.

This chapter will provide an overview of the spatial analytics methods and techniques currently in use or emergent in the game industry or in game research, e.g. for content validation (Chap. 7), as well as methods used outside the games

industry but with a potential for use in the context of games. The chapter will provide a number of guidelines for how to get started with spatial – and spatio-temporal – analytics, based on a range of concrete examples that range from simple visualizations of spatial and temporal data (e.g. heatmaps) to more complex examples, such as overlay analysis and trajectory analysis, e.g. Miller and Crowcroft (2009), Canossa et al. (2011), and Medler and Magerko (2011). Additional resources and methods can be found in Chaps. 7, 12, 14, 18 and 19. The chapter will also explain key concepts and point the way to relevant literature for in-depth exploration.

We will start out with introducing the basics of spatial analytics, and present simple, straight-forward methods which can be applied irrespective of the knowledge of the analyst. We then move on to more advanced techniques such as trajectory analysis and behavior analysis, going through a range of examples from the research literature and indicating how these approaches are useful. Experienced data wranglers are encouraged to skip the introduction sections and jump straight to Sect. 17.4.4 and beyond.

On a final note, in this chapter we will not detail the challenges of collecting and storing spatial telemetry data (see Chaps. 6 and 7), nor the practical aspects of game data mining (Chap. 12), as these topics are covered in other chapters. The focus here will be on the various methods for spatial game analytics and their application to game development and research, e.g. behavioral modeling, player behavior analysis, asset use analysis etc. We are also focusing on player-game interaction telemetry data, not the spatial calculations, that e.g. MMORPG clients, need to perform in order to efficiently process the game state across one or more players or compute responses to player or AI-actions (e.g. finding out which player a mob attacks using nearest neighbor joins etc.), or ensuring low response times, spatio-temporal accuracy of object tracking and so forth. Telemetry data from these kinds of processes are highly useful to monitor how effectively the game client is running and detect flaws/bugs in the system, but are outside of the design-oriented analytics discussed here; for an introduction see Mellon (2009).

17.1.1 Why Spatial Analytics?

In short, there are a couple of key selling points that highlights why spatial game analytics is useful (see Chaps. 18 and 19 for further selling points on visualization of spatial behavior data from games):

- Spatial analytics does not reduce the dimensions of game metrics data, but deals with the actual dimensions of play.
- Spatial analytics allow us to perform analysis not possible in non-spatial analytics providing insights that could not otherwise be obtained, e.g. trajectory analysis.
- Visualizations of spatial game metrics are usually intuitive and actionable insights are readily drawn from them.

- There are decades of research in real-world spatiotemporal analytics available that can be adopted to game analytics. In principle there is little difference between the real world and a game environment when it comes to spatial information, and for all intents and purposes, many games can be considered to be spatio-temporal information systems in the sense that they collect and utilize spatial and temporal data.

There are two important lessons that we learned through our experience working with this analytics:

- Spatiotemporal data can be very complex, selecting the most simple data representation leads to the most general models. Start simple and add domain knowledge where it is available.
- The established data mining methods are very general and can be used for game data as well. However the key of successful knowledge discovery is to design a complete process, starting with clear and useful target and ending with clear evaluation metrics. Furthermore, improving data quality is in most cases much more important than selecting and tuning the data mining algorithms.

17.2 The Basics of Spatial Game Analytics

Given the inherent spatial nature of gameplay, the vast majority of features (or variables) that is measured about player behavior can be attached to spatial and temporal information one way or another. In some games it may not make sense to use specific coordinates, but rather information about which zone or area the player is in when a given event occurs. For mobile games, it may make more sense to analyze the spatial coordinates from the position of the player in the real world. Irrespective, these are all examples of spatial information that come attached to specific player behaviors.

17.2.1 *Spatial Information and Associated Variables*

As mentioned above, spatial analytics in the context of games usually means analyzing player behavior as a function of spatial movement ([Drachen and Canossa 2011](#)). For example, the X, Y, and Z coordinates of the location of a player, as well as the time, whenever that player dies, fires a weapon, accepts a quest, punches an opponent, etc. – i.e. when an **event** of any type occurs in the game.

Four types of information can be logged whenever a player does something – or is exposed to something – in a game: Who is it happening to? What is happening?

Where is it happening? At what time is it happening? More formally, a data object describing a player avatar in a virtual environment can be described by four types of information:

1. Physical attributes of the **avatar** (or avatars – there are various ways in which the player/-s can be visually expressed in games) and the abilities that it encompasses, e.g. for a typical MMORPG class, level, health, strength, speed and similar.
2. The involved **event** or action, e.g. the used abilities, its target; or a chat log tied to a location, chat message to the support system, etc.
3. The **spatial** position, movement speed and current direction of the avatar. In case of games where no player avatar exists, only the second category comes into effect (e.g. position and direction of swipes on a touch screen).
4. All of these types of information can change as a function of **time**, adding a temporal component. We can refer to these components of player-derived telemetry data as **Avatar**, **Event**, **Space** and **Time** information.

A potential fifth component is **Social**, i.e. “to whom”, however the social aspects of telemetry analysis is less well researched in the spatial domain and therefore not discussed further here, but given the application of social network analysis in e.g. social online game analytics appears important (see e.g. Chaps. 4 and 19).

17.2.2 Representations of Spatial Information

Spatial information generally comes in three types: **points** (coordinates), **lines** and **areas** (polygons) (Longley et al. 2005). Point-based data are typically defined using coordinate sets (X, Y and possibly Z). For example, the location a player avatar/character is in when shooting and killing an opponent in a *Team Fortress 2* team deathmatch; or the location where a player or tester reported a malfunctioning texture. Lines are composed of multiple points, but like points have no areal extent. Lines can integrate information about direction. An example could be the line describing the path a player takes through a level in *Gears of War 3* (Microsoft Game Studios, 2011), comprised of many small lines connecting the position of the player logged once per second. Area data have a spatial distribution, for example a building or area with a particular type of vegetation. Note how the spatial measures never stand alone – we measure the spatial component in relation to some meaningful variable, e.g. the path of a particular player, the position of players using particular weapons, the areas where specific environmental situations are in effect, etc. In other words, space is always measured on conjunction with an event (and usually also time). This also means that when dealing with games, such as FPS, player behavior can be mapped as: (1) A spatio-temporal trajectory (path); (2) the sets of actions performed by or performed on the player, each associated with a time stamp and spatial information (Breining et al. 2011).

17.2.3 Tracking Variables – Types of Telemetry

Importantly, telemetry data logging does not need to be triggered by a player doing something. There is an equal value in tracking and logging events, space and time information from the computer-controlled agents of the environments (mobs, agents) as well as virtual objects. For example, logging the spatial behavior of mobiles (entities) allows us to evaluate whether their pathing routines are performing as expected or if any bugs are occurring (dynamic object tracking). Similarly, logging the amount of damage a player does in a fight should ideally be paired with a log of the amount of damage received by the opponent, to check for imbalances or bugs. Currently, the majority of spatial analytics performed in games is done using player-derived telemetry data, and this is also the focus of this chapter, but it is important to remember that play experience is shaped by the interaction between the player and the game – the system side therefore should also be analyzed. This covers, for example, the behavior of computer-controlled entities (MOBs), AI routine monitoring, and all other kinds of behaviors that is initiated by the game software, either autonomously or in response to player behavior. We refer to these two different sources of data as **Player telemetry** or **System telemetry**. These are not the only kinds of game telemetry – there are numerous sources for these, e.g. server performance, tracking user hardware configuration, production-side metrics, but in this chapter the focus is on player telemetry.

Once telemetry data have been obtained, a whole plethora of techniques can be applied to them, the selection of which depends on the goal of the work. The fundamentals are described in Chaps. 12 (data mining) and 14 (fundamental approaches to working with game telemetry data). A relevant example is the heatmap. Heatmaps are basically frequency maps – they show how often a particular event has occurred – traditionally a player death event – on a fine-meshed grid, overlain a map of the game level in question (see Figs. 17.1 and 17.2). A color ramp is used to determine the color of any grid cell, for example red for many death events, moving through orange and yellow and down to green for cells with few death events. The extent of the calculations that has been performed on the data is limited to adding up the events. Heatmaps are, therefore, an example of a very straight-forward – but highly useful – data analysis and visualization technique. Heatmaps have other applications beyond static frequency maps, and are discussed in more detail below.

17.2.4 The Real World vs. Games

Spatial and spatio-temporal analytics are fields of research that have a strong history within a range of sectors outside of the games industry and academia. Geographical and temporal data are vital within the Geosciences, urban planning, environmental research, marketing, population research, Information and



Fig. 17.2 (a) Menu heatmap of user clicks from the game *Youda Jewel Shop* by Youdagames (www.youdagames.com), rendered using *Playtomic's* heatmap function (www.playtomic.com). Interface heatmaps can appear less interesting than gameplay heatmaps, but can provide important information. For example, the menu heatmap showcased here according to the developer showed that people click the user name to change their profile, while there was originally only a non-functional text and a separate “new user”-button. This was changed to the text on the button read “change user” instead of “new user”, and the same functionality added on the “Welcome [PlayerName]” area as well (© Youdagames, courtesy of Youdagames). (b) Heatmap of user clicks from the 2D game *Stackwick Legacy* (Youda Mystery Series) by Youdagames, rendered using *Playtomic* (© Youdagames, courtesy of Youdagames), (c) Heatmap of death events from *Replica Island* (www.replicaisland.net) (© Chris Pruett, courtesy of Chris Pruett)

Communication Technologies, and many other areas, where spatio-temporal information systems are used. In these sectors, spatial analytics has been used for decades and as a result a massive number of approaches and methods are available (Koperski and Han 1995; Ester et al. 1998; Han et al. 2001; Shekhar and Huang 2001; Shekhar et al. 2003).

In principle, however, there is little difference between the real world and a game environment when it comes to spatial information, and for all intents and purposes, many games can be considered to be spatio-temporal information systems in the sense that they collect and utilize spatial and temporal data (*event*, *space* and *time* data). It may not be immediately obvious what for example a navigation system for the transportation sector and an RPG has in common, but there are many similar challenges – e.g. finding the fastest or least resource-demanding trajectory. This is



Fig. 17.3 Combining chat logs or bug reports with the spatial location of the player provides the ability to pinpoint problems. Employing data mining on a word cloud provides the ability to filter incoming messages and highlight key words, e.g. “bug” or “how do I” in the MORPG *Star Wars: The Old Republic* (2011, Electronic Arts) (© Bioware; courtesy of Bioware, reprinted from Zoeller (2011) with permission)

most obvious when comparing a 3D persistent world environment, e.g. MMORPGs such as *World of Warcraft* (Blizzard, 2003) or *Lineage II* (NCSOFT, 2004), to the real world. In these games, a player navigates a 3D environment filled with other entities and objects, much like we navigate our real-world environments. The chance of meeting an orc horde is somewhat smaller in the real world, but the chance of meeting a horde of travelling salesmen comparably higher. The point being that from the perspective of spatial analytics, it does not really matter whether an environment and the entities and objects that populate it are real or not, and this means that that at least some of the body of knowledge built within the sectors employing spatio-temporal information systems is applicable to games (Kriegel et al. 2011; Drachen and Canossa 2011).

In some aspects, telemetry data being derived from games makes analysis even simpler. In real-world monitoring and environmental tracking systems the measured data is often uncertain and error prone. Furthermore, position or state updates are usually much sparser. Tracking the position of a car via GPS has a 10 m localization error. Furthermore, using GPS positioning and transmitting data consumes a lot of energy making online tracks often very situational. Telemetry data does not suffer from these effects. A gaming server knows exactly where its objects are at each point of time (barring bugs or flaws in the system). To conclude, analyzing game data

can usually rely on much higher data quality than real-life data collected from e.g. satellites or sensors, which makes it easier to find significant information. Figure 17.3 shows an example where text mining has been used to find the exact locations where players in the in-game chat of specific key words occur, signifying a bug or other user-related problem.

A stellar example for a technique that transfers well from real-world spatial analytics to game spatial analytics is heatmaps (again), having a long history in e.g. environmental research, e.g. Ghanem et al. (2004). It is worth noting however, that the goal of spatial analytics in games vs. real-world contexts can be different, necessitating modification of how particular methods are employed. For example, in games a key aspect of design is ensuring player engagement, not necessarily finding the fastest path through a particular area. This, however, does not impact on the usefulness of spatial analytics.

17.2.5 Tools for Spatial Game Analytics: Geographic Information Systems

There are a wide variety of tools available for performing spatial analytics. These range from major integrated packages that offer a variety of ways to work with and visualize spatial data (e.g. ArcGIS, MapInfo), to smaller tools focusing on particular forms of analysis or for integration in specific environments or specific application areas (e.g. Tableau). There are also a number of open-source tools available (e.g. QGIS), which may be easier to adopt than developing a solution from the bottom up. The easiest way to locate an open source toolbox useful to a particular spatial analytics task is to figure out what type of analysis is required, and then use this information as the search parameter. Because these tools are developed outside of game development, the barrier of entry in terms of adopting and adapting them to games work can be relatively high. Currently, most major game publishers have their own custom-build telemetry systems which can also handle spatial analysis and visualization to some degree, although the specifics of these remain largely unrevealed (e.g. Microsoft's *TRUE* system, see: Kim et al. (2008)) or Ubisoft's *DNA Viewer* (Dankoff 2012).

Developed specifically for spatial analytics, **Geographic Information Systems** (GIS) (Fig. 17.4) form the major platform for data management systems used to all or some of the following: capture, store, manage, retrieve, analyze, query, interpret and display spatial and associated temporal information in the form of e.g. maps, reports and charts (Longley et al. 2005). A GIS is, thus, similar conceptually to the data management systems used in games, which control objects and entities with specific attributes and behaviors inside the game environment. The difference is that a GIS is specifically designed for the management of spatial and spatio-temporal data.

In a GIS, map features are linked with attribute information. For example, a level map is linked with locations of quest providers, different types of environments,

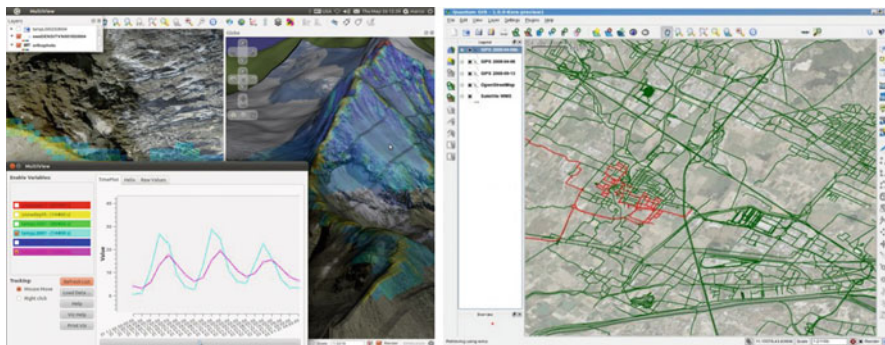


Fig. 17.4 A GIS working environment (here Quantum GIS, an open source GIS). The main window displays the current map, features and analyses running, using the *left* and *top* menus. *Left* image displays a time plot generated over a map, the *right* image a street map (Images used with permission from the QGIS Project, www.qgis.org)

player trajectories, or other spatially distributed features. When working in a GIS environment, game telemetry can be referred to as **geospatial metrics** – this signifies that the data have a spatial (points, lines, areas) as well as a thematic component (number of kills, environment type, bot density, etc.).

In a GIS, player behavior metrics derived from telemetry data (also referred to as “gameplay metrics”, e.g. (Drachen and Canossa 2011)) can be visualized on top of the game environment – whether 2D maps, screenshots or directly inside the game environment via linking with the game engine. The data can then be analyzed. For example, 500 player trajectories can be imported, plotted in on top of a level map, and then analyzed in terms of the speed of progression through the game’s environment (and whether this matches design expectations). Most GIS packages permit interactive maps to be produced. These are prepared by an analyst, and then exported to e.g. a website. From this the interested stakeholder can dynamically select or unselect specific layers or visualizations. It is also possible to divide maps into subsections and name these differently, etc. There exists on the Net various small free packages for generating visualizations, but usually these are static.

When mapping spatial game metrics onto maps that themselves can contain detailed feature information, a high degree of flexibility in terms of spatial and spatio-temporal analysis is gained. For example, when calculating the number of kills occurring “inside” a specific type of environment vs. “outside” in a FPS – as well as specific numbers for each environment. Multiple datatypes can be added on top of (or as) maps, for example in the form of multiple layers each containing one feature. A GIS allows calculations to be performed along as well as across feature layers and their attributes (see Fig. 17.5 for an example where multiple layers of behavioral features (player trajectory, location of NPCs, death events, disguise use and ideal path) are added on top of a map (terrain layer) from the FPS/sneaker *Hitman: Blood Money* (Eidos Interactive, 2006)).

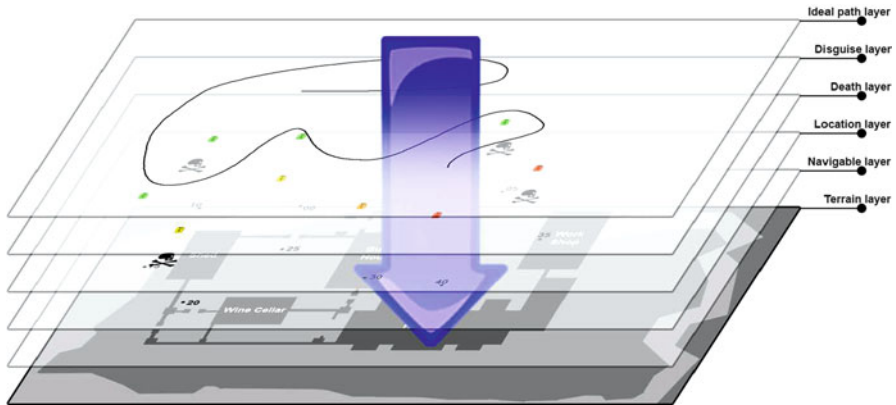


Fig. 17.5 A GIS represents different data sets as layers on top of a game level map, and provides the ability to perform calculations across them (often overlay analysis, but can also be drill-down processes (Chap. 12)) (represented by the *blue arrow*) (Reprinted from [Drachen and Canossa 2009](#) with permission; image is © 2009 by ACM, Inc.)¹

17.2.6 Middleware for Game Analytics

There exists a variety of middleware solutions for the logging and transformation of game telemetry, mostly formed by relatively new startups, e.g. *Game Analytics*, *Playtomic*, *Playnomics*, *Honeytracks*, etc. Some business intelligence providers like *Plateau* are also moving into this area, as are engine developers. The specifics of the solutions offered by each company varies, but generally provide a system for adding hooks into the game code, which ensures that relevant telemetry data are transmitted to a server-side database, which communicates with a web-based front end through which the user interacts with the data (see text box). Telemetry data are during the process transformed into game *metrics* – e.g. data on player ID and session time combined to generate an average completion time metric. When it comes to spatial analytics, the tools offered by these companies remain in their infancy compared to the commercially available packages available in other sectors, e.g. geographic information systems. In-house systems for game analytics appear to exist in most major publishers and developers. Examples include the *TRUE* system at Microsoft Game User Research (Kim et al. 2008; Ramero 2008), *Playtest DNA* at Ubisoft Dankoff

¹ACM COPYRIGHT NOTICE. Copyright © 2009 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481 or permissions@acm.org

(2011, 2012), *Amber* at Square Enix and *Skynet* at Bioware (discussed in Chap. 7). In the public research domain a few tools have also been developed, e.g. Ben Medler's *Data Cracker* (Medler et al. 2011) (discussed in Chap. 18), Andre Gagné's *Pathway* (discussed in Chap. 19) and Pedersen/Canossa's *G-player* (info on all of these and others can be found on the Net).

17.3 Directions in Spatial Data Mining

The purpose of data mining (or knowledge discovery) is to automatically extract patterns on data sets – patterns that are statistically correct and potentially useful (Chap. 12). In this context the expression pattern is extremely general and ranges from complex mathematical functions, over rules to frequently occurring constellations. The knowledge that can be extracted from spatial information is usually more complex than the general patterns and prediction rules employed in ordinary data mining and machine learning. Thus, extracting spatio-temporal patterns often requires specialized methods to consider the spatial relations within the data.

In general, a data object describing a player or another object in a virtual world can be described by two types of information. The first are attributes like health, strength, fire power, class, etc. However, since this object is moving in a spatial environment, it also can be described by a position, its movement speed and its current direction. Also, all types of information might change over time adding the temporal component to our view on the data. Spatial data mining methods explicitly consider the spatial part of the data and treat them in a different way than the ordinary features describing the object characteristics. Thus, it is possible to explicitly distinguish information about which objects are located close on the map and which objects are similar based on their other attributes. Finally, spatio-temporal data mining allows considering the change of both types of information over time. For example, for describing the behavior of an avatar collecting herbs in an MMORPG like *World of Warcraft*, it is required to consider both aspects. The character must be skilled in herbalism with enough skill points to pick herbs spawning in the area the player is currently moving. Furthermore, the movement in the virtual environment gives major indication about his searching behavior. If the character just stops to pick up a herb and then continues to a dungeon or a city, the main purpose on the movement might be travel instead of collecting herbs.

There are several classical tasks in spatial data mining that were motivated by real-world problems in geography, biology or medicine. These broadly correlate with specific categories of spatial data mining/analysis methods. While methods for data mining in general were introduced in Chap. 12, the categories mentioned here specifically focus on applications to solve spatial tasks (the list is by no means inclusive, but serves as an overview).

17.3.1 Spatial Predictive Models

Spatial predictive models find spots on a map where certain events might occur. In general, predictive models learn the characteristics of locations where certain events might occur. By applying machine learning techniques, it is now possible to predict the likelihood that this event might occur on a given location. Example use cases for this data mining task are predicting crime spots, spots for natural disasters, like flooding, or predicting the breeding spots of birds. An application in games may be to predict the location for building a city or base in a strategy game. Though general classification methods might be applicable to the task, there are specialized techniques for training a spatial prediction functions like Markov random fields (Li 1995; Shekhar et al. 2002) and spatial autoregressive models (Shekhar et al. 2002). Both methods are based on statistics and include the characteristics of a certain area in combination with the surrounding area in the environment. For example, a location for a city in *Civilization V* (2K Games, 2010) might be good on itself because there are plenty of farming opportunities and a river for added gold income. However, there might be other spatial factors like a large distance to the next friendly town or enemy fortifications being close by. Spatial predictive models can assist with predicting the best place to build cities given the defined constraints.

17.3.2 Spatial Outlier Detection

Another classical task in spatial data mining is spatial outlier detection. In general, an outlier is defined by a data object which does not fit to the data distribution holding for the rest of the data set (Hawkins 1988). Spatial outlier detection specializes this concept by comparing an object to its spatial neighbors. For example, a house having a moderate price in the middle of a high class residential area yields an interesting spatial outlier. In this example, the spatial component is essential to the interestingness of the object because there might be other estates having a similar price. In computer games, spatial outlier detection can be used to find exploitation spots. In particular, there might be spots on a map that allow for attacking computer opponents without having them defending themselves. Finding spatial locations where avatars were never harmed surrounded by areas where attacks occurred quite often will hint to exploiting behavior. Example techniques for spatial outlier detection are described in Lu et al. (2003).

17.3.3 Spatial Co-location Patterns and Spatial Rule Mining

The next data mining technique being specialized on spatial data is deriving rules about spatial co-location and other relationships between spatial locations. In the case of spatial co-location rules (Shekhar and Huang 2001), we are interested

in pairs of certain objects occurring nearby each other. A general application in biology is the co-existence of certain animal species. In games, spatial co-occurrence rules can be used to derive higher order tactical elements like the group compositions of units in a strategy game. For example, analysis of this type might indicate which type of compound force of units that can grouped together for building exceptionally strong attack or defensive armies in strategy games.

Another method resulting spatial rules is spatial trend detection. Trend detection is concerned with the change of features when moving into a certain direction. For example, a spatial trend could indicate increasing housing prices when moving from a low class suburb to the uptown city center. Methods for spatial trend detection are described in Ester et al. (1998).

A final task in this category is spatial association rule mining. A spatial association rule describes a probabilistic implication. An example rule in the gaming area might be that 80% of the cities being build closer than 50 km to at least two of another player's settlements won't survive until the end of the game. In general, an association rule is considered a spatial association rule if the predicates being used for the formulating the rule describe spatial relationships or characteristics. An algorithm for deriving spatial association rules can be found in Kopersky and Han (1995).

17.3.4 Spatial Clustering

Spatial clustering describes the grouping of spatial objects that are similar with respect to their general characteristics and additionally, are positioned close to each other on the map. In general, objects from different clusters should be as dissimilar as possible. A general application of spatial clustering is to find out whether spatial objects are uniformly distributed over the virtual environment or if there are hot spots where certain events happen more often. An example is clustering the spatial locations of criminal activity. The result describes areas where a certain type of criminal activity occurs more often than in its surrounding. Spatial clustering in game data can be employed for several tasks. For example, clustering the avatar position at a certain point of the day offers useful input for load balancing in a shared virtual environment. Other applications might be to check whether people in a first person shooter or MMORPG prefer to fight their battles on a certain dedicated location. An overview to clustering algorithms, which are suitable for spatial data can be found in Han et al. (2001).

There are several more recent directions in spatial analytics that are closely related to the methods being already employed in the area of spatial game analytics. These will be discussed in connection with case examples (solutions) below.

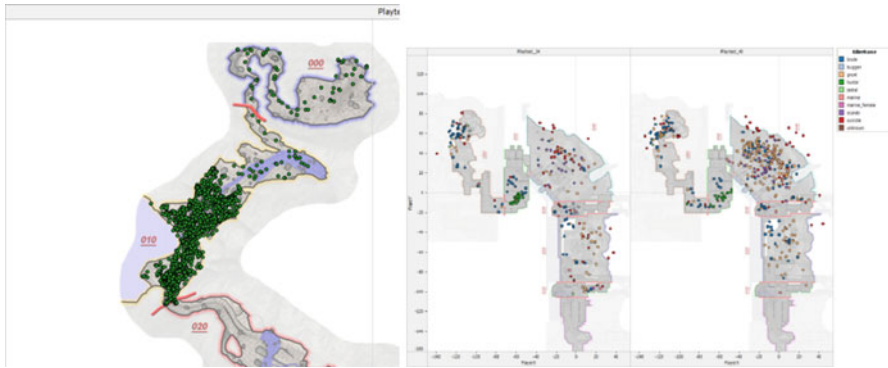


Fig. 17.6 Examples of univariate spatial visualizations: Two visualization of playtest data from Microsoft Studios Research. *Left*: plots of player progression through a level from *Halo 3* after a specific amount of playtime; *right*: plot of death positions from *Halo 3*, color coded according to the cause of death (different forms of enemies, suicide or unknown causes) (© Microsoft Studios Research; used with permission from Microsoft Studios Research. Reprinted from Romero (2008))

17.4 Categories of Spatial Game Analytics

Having covered the basics of spatial analytics, this section will focus on practical examples and the insights that can be gained from different types of analysis. The section will start out by introducing some of the simplest forms of spatial data visualization, like heatmaps, gradually progressing towards more advanced methods. Some of the examples are already well known in game development and research, others are more novel, being inspired by various applications of spatial analytics outside the games sector which can be adapted to game development. There are outside of game development literally thousands of methods to visualize and analyze spatial data, depending on the purpose and field of work (Demers 2008); however, in the following, the focus is on examples of methods already employed in or of direct relevance to game analytics.

17.4.1 Univariate/Bivariate Map Analysis and Visualizations

As mentioned above, spatial game metrics can be visualized via synthesis without an accompanying analysis phase operating on the data. Univariate (one variable) and bivariate (two variables) visualizations are among the most commonly seen in the game development and game research literature, e.g. Ramero (2008), Kim et al. (2008), Drachen and Canossa (2009), and Zoeller (2011), and are relatively easy to generate and intuitive to interpret (e.g. Fig. 17.6, which shows point features overlain a game level map from *Halo 3*). Visualizations of this type has also been generated

by players, e.g. with the goal of providing guidance or strategic advice to other players, as is commonly seen in mapping of resources from MMOGs and open-world games such as *Fallout 3* and *Oblivion*. Developers similarly generate visualizations not only for internal use, but also for the player community, e.g. in the form of global heatmaps for *Halo 3* (Bungie 2011).

Similar visualizations are used internally to evaluate designs – e.g. analyzing if any mobiles wander outside the area they are supposed to stay within, where players pick up items, and so forth. Just plotting where something happens on a map provides a context that a non-spatial analysis will not provide (Fig. 17.6). Spatial visualizations provide a means of evaluating gameplay telemetry (gameplay metrics), in the context that the games are actually played in, adding an explanatory dimension that is not available in non-spatial analytics (e.g. causes of death as a function of location; Fig. 17.6).

17.4.2 Heatmaps

Heatmaps are similar to the single-feature visualizations described above (Figs. 17.1 and 17.2), but go one step further in terms of performing manipulations on the aggregated spatial data. Heatmaps originated in 2D displays of the values in a data matrix, but have since then been adapted to a variety of contexts, from heatmapping of eye gaze on websites to mapping of environmental factors. The examples are numerous, and there is a substantial amount of experience available from these application areas which are applicable to game analytics. For a guide on how to generate simple heat maps for games, see Pruett (2010) and references therein.

A common way to generate a heatmap is to divide the game area under scrutiny into a grid of cells, and sum up the number of events that occur within the area covered by each cell. Adding a color ramp (e.g. green to red) allows easy interpretation of the data by highlighting areas of maximal or minimal activity (as show in Figs. 17.1, 17.2 and 17.7) (Drachen and Canossa 2011). Other approaches to generate heatmaps employ density kernel functions, emphasizing the importance of regions with high density of events, or interpolation in order to attempt simple prediction in areas that there are no events registered for. Different visualization methods are optimal in different situations.

Heatmaps are in games most commonly used to aggregate and visualize death events, forming maps of the lethality of a specific game area. This is a highly useful form of visualization in 3D action/shooter games like *Quake* (ID Software, 1996), *Unreal* (GT Interactive Software Corp., 1998), *Half-Life* (Valve, 1998) and *Team Fortress 2* (Valve, 2010), in single-player as well as multi-player mode. Heatmaps show designers exactly where players and conversely bots die and how frequently. Heatmaps showing death events caused by specific weapons, enemy types or similar have been developed for example *Halo 3*. Analyzing the spatial distribution of different causes of death adds a second layer of usefulness to heatmaps, as it is thus possible to evaluate the impact of different enemies or weapons in different regions of a game map.

Fig. 17.7 A heatmap of all of the failures of the playtest participants in *Assassin's Creed Brotherhood*, in the fifth mission of Sequence 5. The mission was subsequently made a little easier. (Legend: Points go from green to red based on density) (© 2007–2012 Ubisoft Entertainment; courtesy of Ubisoft. Reprinted from Dankoff (2011))



Most game heatmaps showcased on the Net and in conference presentations have been 2D, i.e. using the X and Y coordinate of death events. However, applications for generating 3D heatmaps are emerging and these provide the ability to evaluate map lethality in three dimensions, which adds yet another useful feature to games with 3D setting. In these games, 3D heatmaps allow for better interpretation of the effect of level design, and helps alleviate the errors that can occur when data from multiple Z-axis levels (i.e. height, e.g. a building with two floors) are layered on top of each other, thus falsely indicating that a particular ground-area level is more lethal than it actually is. A guide for how to compute heatmaps can be found at: <http://blog.corunet.com/how-to-make-heat-maps>.

Using heatmaps requires no special training, and allows for quick identification of e.g. bottlenecks, but it is important to ensure that the heatmaps are highly granular, to evaluate for example if a particular spot is too well or too little covered. Heatmaps are also useful as an indication about whether any area in the map is not being used/experienced, although trajectory data are more precise for this purpose (Romero 2008).

Importantly, heatmaps are not limited to the mapping of death events or to the mapping of point-based data. The simple aggregation process of generating heatmaps means that e.g. player trajectory data (line data) can also be used as the basis for heatmap visualizations, showing in these cases the density of chosen movement paths. Similarly, area data (polygon data) can be aggregated. In general, most telemetry features (variables) that have a spatial component can form the basis for heatmaps. The heatmap visualization is simple and flexible, and forms an excellent starting point for getting to grips with spatial game analytics.

Finally, while not a subject that has been discussed much in game development at the time of writing this chapter, combining heatmap visualizations with the temporal dimension of telemetry data adds a dynamic quality and allows for a better

Table 17.1 This table shows the number of kills done by the playtest participants in sequence 4

Player	Total time played	Total kills	Regularkill	Counterkill	Combokill	Stunkill
ACB0016SINGLE	00:38:02	48	40	0	4	4
ACB014SINGLE	00:44:30	44	28	1	7	8
ACB015SINGLE	01: 17:05	65	44	1	8	12
ACB09SINGLE	00:18:57	6	5	1	0	0
ACB10SINGLE	00:29:14	42	6	4	21	11
ACB12SINGLE	00:29:39	12	10	0	0	2
ACB13SINGLE	00:38:24	37	20	0	9	8
Average	00:39:24	36.29	21.86	1.00	7.00	6.43

It allowed us to confirm the design intention that players should use the counterattack less often, in favour of the new combo system (© 2007–2012 Ubisoft Entertainment; courtesy of Ubisoft. Reprinted from Dankoff (2011))

understanding of the flow of gameplay/the playing experience (see however Gagné et al. 2012). Essentially, heatmaps combined with temporal information allow designers to follow how the mapped events occur over time, and it is not difficult to think of situations where it is interesting to know for example what players do the first 30 s on a map as opposed to a simple aggregate of the sum of their activities. Temporal (dynamic) heatmaps have been mentioned in relation to Unreal Engine 3 (the Master Control Program) and in academia by Drachen and Canossa (2009) and Canossa et al. (2011), but published examples from application to games remain rare (Table 17.1).

17.4.3 Multi-Variate Visualizations and Overlay Analysis

Visualizing and analyzing multiple spatial features (variables) enables evaluation of how these variables interact (Demers 2008; Drachen and Canossa 2009). For example, combining heatmaps with other spatial data, for example trajectory data from player movement or projectile paths, provides for highly detailed analysis of the dynamics of a playfield. Two or more heatmaps from different builds of a level can also be overlain (see below) and the effect of design changes evaluated. Similarly, mapping two heatmaps generated from different variables (e.g. kills and deaths) provide even more information, e.g. whether players kill and die in the same spots or if there are e.g. areas that are much more likely to be places from where players kill other players.

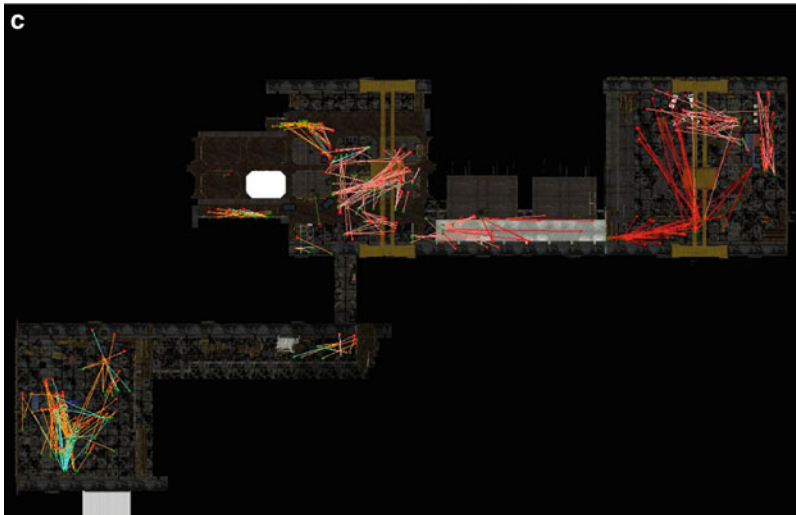
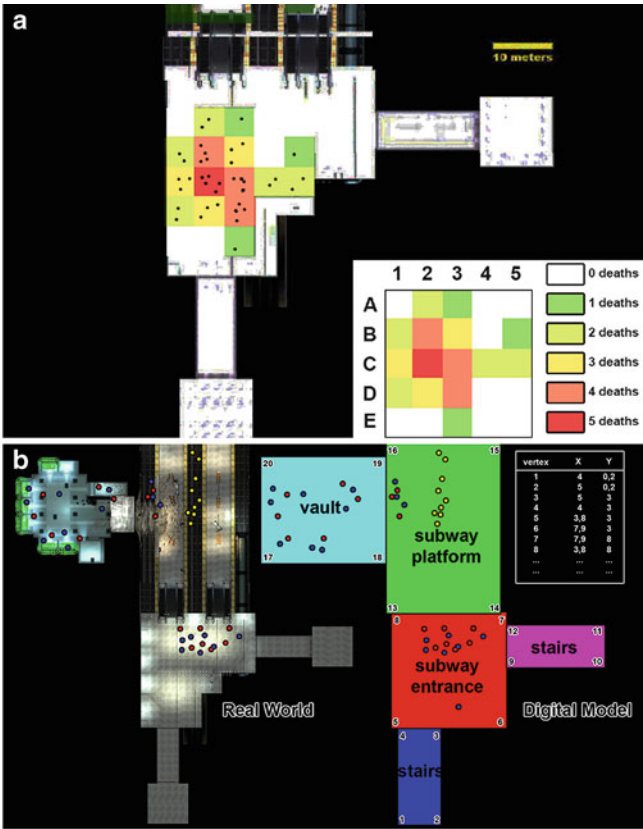
Analyzing and visualizing multiple variables can in the context of a typical GIS package like ArcGIS approached via a process known as **overlapping**. This mapping principle has also been applied for game analytics by Drachen and Canossa (2011). Overlapping is a spatial operation in which two or more maps or layers registered to a common coordinate system are superimposed on top of each other (Demers 2008), and operations then performed on these layers. The overlay operation thus creates composite maps by combining datasets. The purpose is to visualize

and analyze the relationship between features occupying the same space, e.g. a map of a game level, player trajectories logged for that level, and death events (example in Fig. 17.8). The overlay function can be based on simple operations such as synthesis (trajectories + where players die), or analysis (e.g. multiplying, subtracting, find averages or co-occurrences – for example subtracting two heatmaps from each other to evaluate the difference between them) (Fig. 17.9 provides an example where a player kill heatmap and a player death heatmap are subtracted from each other to provide a “balance” heatmap). Drachen and Canossa (2011) describes an example of overlay analysis of areas where players die of different causes; to locate areas of a level in *Tomb Raider: Underworld*, where multiple threats to the players are present, correlating this information with death events to find that areas with multiple threats are more lethal than areas with a single type of threat.

As mentioned above, spatial data can be points, lines or polygons (areas). In turn, these can be displayed in two basic formats: raster or vector models. **Raster models** are grids – each grid cell contains information about one or more variables (usually one) (Fig. 17.8a); **vector models** are comprised of point and lines (Fig. 17.8b). Heatmaps are based on raster models. Raster and vector models differ significantly in the way overlay operations are performed, and generally overlay operations are performed most effectively in raster-based models. A GIS is usually able to convert raster models to vector models and vice-versa, allowing for vector models to be converted to raster models, overlay operations performed, and the result converted back to a vector model.

Overlay operations can be used to perform a variety of analyses. For example, flow maps are visualizations of direction, and can be used to show main directions of player navigation through a playfield (whether 2D or 3D). Accounting for the temporal nature of behavioral data adds another layer of knowledge to overlay analysis. In summary, multi-variate synthesis and analysis provide a good starting point for understanding how people play a map and to at least some degree infer why, thus providing a source for feedback on game design.

Fig. 17.8 (a) Example of how a raster-based heatmap is comprised by a grid of cells, within each the total amount of events occurring (here death events) are summarized. Illustration by Alessandro Canossa. (b) Basic example of the construction of a polygon-based model of a “real world” (i.e. the real game environment) on top of a game level map (digital model). Locations of player death events forms a third layer (point events). Combining the polygon-based model and the death events layer provides the ability to perform calculations across these two layers, for example to calculate how many death events that fall into each of the major areas of the game level. Illustration by Alessandro Canossa. (c) Location and weapon use in *Kane & Lynch: Dog Days* (2010, Eidos/Square Enix). The locations of the player whenever he makes a killing shot, has been marked with green dots. The lines show the lines of fire, and the colors signify the use of different types of weapons. Red dots the location of the killed enemies. Such visualizations of game metrics provide game designers with tangible information about the behavior of the players, and highlight any flaws or problems in the game design (Reprinted from Drachen and Canossa 2011; image is © Inderscience Enterprises Ltd.)



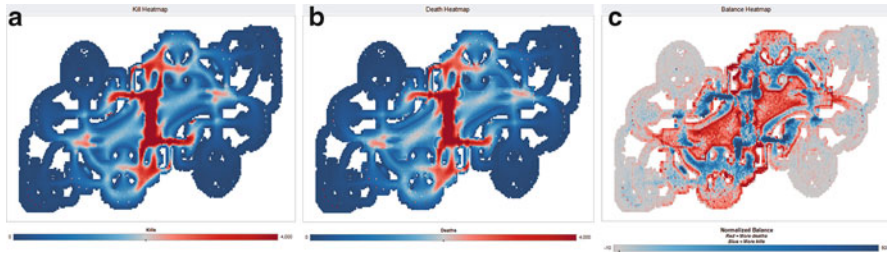


Fig. 17.9 Example of overlay analysis. (a) a heatmap from the “Molten” map of the game *Transformers: War for Cybertron* (2010, Activision), based on a dataset of 30 million rows collected from public Team Deathmatch games. (b) a heatmap showing the positions of players at the time they killed another player. (c) A balance heatmap generated using an overlay function, where the death heatmap is subtracted from the kill heatmap. Areas with negative values (*red*) indicate dangerous areas, areas with positive values (*blue*) indicate areas that are safer. An interesting conclusion is that wall areas appear to be dangerous, possibly due to restricted movement (© Sean Houghton; Reprinted from Houghton (2011) with permission)

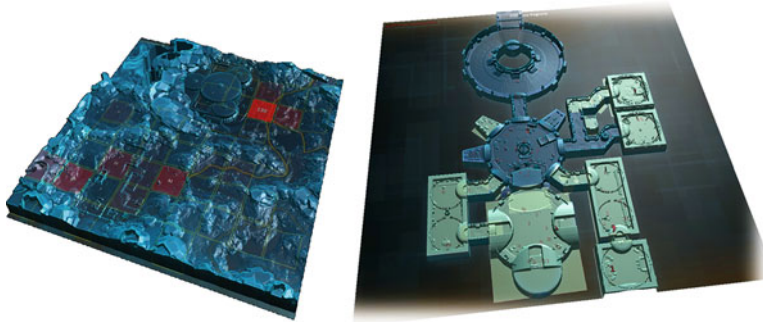


Fig. 17.10 3D visualizations of spatial telemetry data from *Star Wars: The Old Republic*. Left: Compressed Z-axis map with multiple layers of data. Right: Map from an instance in the same game, with a region phased and highlighted. Numbers/areas indicate volumetric triggers (© Bioware; courtesy of Bioware, reprinted from Zoeller (2011) with permission)

17.4.4 Visual Dimensions: 2D vs. 3D

Spatial visualization of telemetry data need not be confined to two dimensions. Many games are 3D, and in these games there are generally variations in the Z-plane, leading to different opportunities for players (e.g. elevated positions). Levels may even have multiple planes on top of each other (e.g. floors in a building), or tunnels underneath the surface (Fig. 17.10 shows two examples multiple planes in game levels, from *Star Wars: The Old Republic*).

Projecting multiple layers of data on top of each other provides bias, or make spatial analysis incomprehensible, and should ideally be kept separate. Similarly, some visualized patterns may not be clear unless the 3D nature of the playfield is considered – recall that players experience these playfields in 3D, and sometimes

viewing the overlapping areas together is the best way to understand the data. The problem can be solved by using 3D visualizations of the data, something that is still rare in games (but see e.g. Zoeller (2011) and Blackhurst (2011)), but a powerful tool for designers for evaluating player behavior in the dimensional space players experience games in (data can even be visualized inside the game engines).

The advantage of 3D visualization of telemetry data, chat/bug report data etc. directly in the game environments or vector altitude maps provides infinite resolution for zooming comparing to working with bitmap images, and the maps are easier to read, especially for people who are not familiar with the playfields. 3D visualizations also generally permit the overlay mapping of several variables while retaining the ability to clearly understand the data – e.g. it is possible to follow the players around the map and understand how they perceive the environment. 3D maps are also easier to read than 2D maps, as e.g. identifying landmarks is easier (Zoeller 2011).

17.4.5 Trajectory Analysis

A trajectory is a description of the movement of an object in space over a specific period of time. In the case of computer games, for example, a trajectory could be the navigation of a player through a game level. Along the way, various events occur – fights, item pickups, item uses etc. Trajectory analysis is thus useful for getting close to the actual gaming experience. Analyzing trajectories is currently used to e.g. locate illegal bot programs in online games, examine group behavior, study player tactics, asset use – these are just examples, the uses are manifold (see Figs. 17.11 and 17.12 for examples).

The step from spatial to spatio-temporal data mining allows us to consider much more complex events in the real world as well as in the field of computer games. The most common type of spatio-temporal data objects is a trajectory describing the movement of an object in space over a certain period of time. Though it is possible to consider trajectories as spatial objects only, it often loses important information of an event. In other words, the information that two people followed the same path to travel from one location to another is different from the information that they travelled together. In computer games, strategic behavior often requires that the members of a team mutually support each other. However, in FPS and MMORPGs supporting each other often implies a certain spatial closeness. Thus, a single pinpointed attack of a large group of units is often very different from several separated attacks of small groups. When just considering the absolute trajectories both tactics will appear very similar. Only if we additionally consider temporal closeness as well, we can distinguish one tactic from the other.

An aspect adding to the complexity of analyzing spatio-temporal data is the fact that cooperative behavior is usually limited to a certain period of time. In other words, players might coordinate during an attack, but after the battle is fought teams might break and regroup in different constellations.

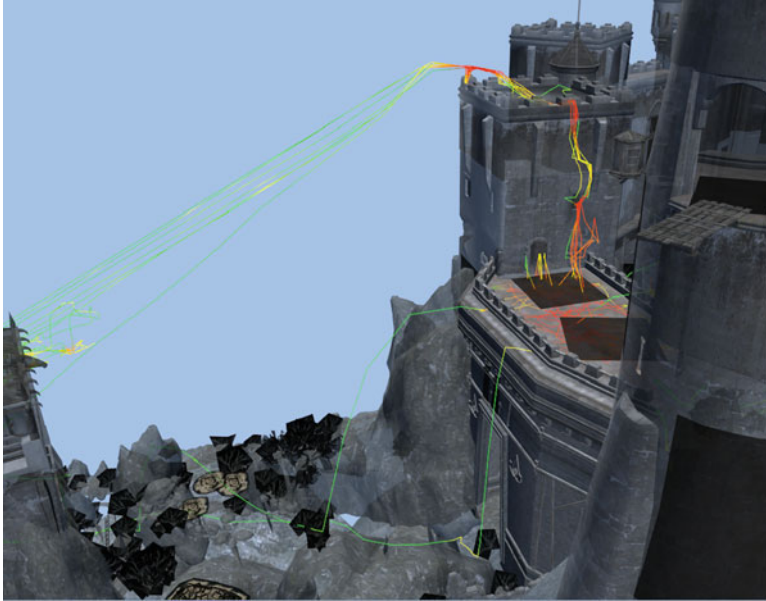


Fig. 17.11 Trajectory visualization developed by the user research team at Ubisoft Montreal using their native system for logging, analyzing and visualizing player telemetry: DNA Viewer. This image shows the trajectories of eight players. Six of them are shown parachuting from a castle tower to another tower as intended by the game design, but two other players making their way down via an unintended path along the castle walls. The visualization was generated based on feedback from a user research team member, who noted that some players were not taking the appropriate path which led to a descent via parachute, but instead descending using an alternative path, which bypassed the tutorial and results in a less fun playing experience. The user research staff at Ubisoft Montreal used the viewer component (DNA Viewer) of their in-house produced telemetry system, Playtest DNA, to display both 2D overview maps, as well as 3D representations of the whole space. Using the viewer, 3D maps are taken directly from the developer's production tools (e.g. 3DS Max) and integrated into DNA Viewer, offering a direct reproduction of the game world but with added behavioral telemetry data. Using the visualization, the designer of the level was able to easily fix the area to ensure that the players took the intended path and used the parachute. This resulted in a more streamlined and enjoyable experience for the player (© 2007–2012 Ubisoft Entertainment; courtesy of Ubisoft. Reprinted from [Dankoff \(2012\)](#))

Mining trajectories has attracted the attention of researchers and practitioners in various fields such as GIS systems, surveillance systems or even location based search engines. A driving factor for the area of trajectory analysis is the availability of data. With the wide spread use of smart phones, GPS-devices, portable computers, surveillance cameras and sensor networks, it is possible to collect information about people's locations and sometimes even about their behavior. The information a smart phone collects about its user includes locations, travel speed, telephone connections, short messages and email, search engine queries, personal calendars etc.. With such a large variety of information about a person's behavior, spatial analytics

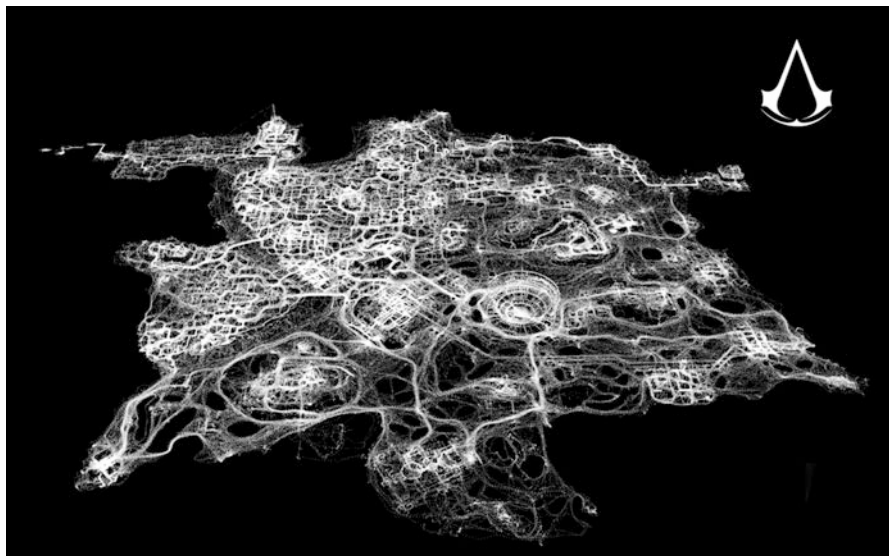


Fig. 17.12 Trajectory visualization developed by the user research team at Ubisoft Montreal. The visualization is from *Assassins Creed: Brotherhood*, specifically the area of the game based on the city of Rome. Trajectory visualization created using 138 tracked players over 10 days after the release, for a total of 8,765,149 points or about 1,200 h of gameplay (© 2007–2012 Ubisoft Entertainment; courtesy of Ubisoft. Reprinted from Dankoff (2012))

currently more and more extends from analyzing movement information to analyze complex spatio-temporal behavior. Furthermore, data mining expands to model the dependency of a person's behavior to the behavior of other people.

Examining social mechanism is a further aspect of some games which has to be considered under the light of spatial information because there are some things in games (and life) that require two people to be in the same location. If we compare the scenario described above to an online game setting, we will see no difference. Since the game server has to manage player movement, interactions, chats and other directions in the game, a game provider can as well record trajectories and interactive actions of players within the game.

As in real life there is no guarantee to capture anything a player does. For example, the game server has no access to voice chats such as *Team Speak* (www.teamspeak.com) which might be used for faster communication. On the other hand, the data recorded in games often shows less uncertainty because there is no localization error caused by a tracking device. Furthermore, player behavior usually serves the purpose of being successful in the game. Thus, the observed actions can be related to a known set of goals and intentions.

In general, the techniques to analyze trajectories and other spatial behavior in geo-information systems are applicable to game data, and, therefore, it is relevant to briefly survey recent trends in the analysis of trajectories. To represent a trajectory a common way is to record the position at a consecutive sequence of points in time,

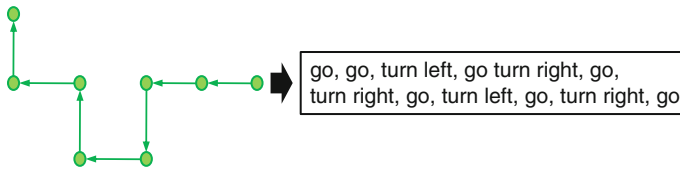


Fig. 17.13 Absolute Trajectory and the corresponding relative movement pattern for simple moving primitives (*go*, *turn left*, *turn right*)

which are called waypoints (illustrated in Fig. 17.4). A problem with this representation is that movement is usually done in a fluent way and not in jumps between the waypoints. To allow for position estimation at an arbitrary point of time, movement between the waypoints is assumed to be linear and with constant speed. In the end, the trajectory representation is always just an approximation having a certain resolution corresponding to the number of recorded waypoints in a certain period of time. Depending on the application this resolution might vary. Nonetheless, it is always necessary to consider the trade-off between the exactness of the data and its volume. In many cases, having too large a resolution causes efficiency and memory problems. Furthermore, high resolution is often counterproductive to many analyzing tasks, because it may obfuscate similarities. Unlike the common setting in the GPS scenario of spatial analytics, trajectories being derived from computer games do usually have a large resolution. Since a game knows the exact position of each object at each tick, game trajectories should be reduced with regard to their resolution in order to increase their comparability and decrease the processing overhead.

Another way to describe trajectories is the change in relative movement. While the method described above using absolute positions can be used to detect regions and paths on the map, relative motion patterns are more concerned with behavioral patterns. For example, it is possible to distinguish Bots and Players based on relative trajectory patterns. Relative motion patterns are often stored as a sequence of movement commands like “*go straight, turn left, go straight, stop*” (Fig. 17.13). Relative motion also has some kind of resolution which has to be chosen carefully. For example, considering the exact angle of a turn has a higher resolution than only distinguishing “turn left” and “turn half-left”. To derive relative motion patterns in games, there are generally two possibilities. The first is to derive absolute trajectories first and then derive the relative motion being contained. A second opportunity to acquire relative movement is to directly record the player commands controlling the movement of the avatar. Depending on the given observation this method is often more direct and thus, the received trajectories are better suited to mirror the player behavior.

One of the important forms of trajectory analysis in games is **clustering**, i.e. classification of data based on the degree of similarity and dissimilarity. The basic goal of clustering trajectories is to find groups of trajectories being as similar as possible while trajectories belonging to different groups should be as dissimilar as possible. Clustering can be based on just trajectories or trajectories + event data (e.g. where players use specific abilities, pick up items, interact with other players).

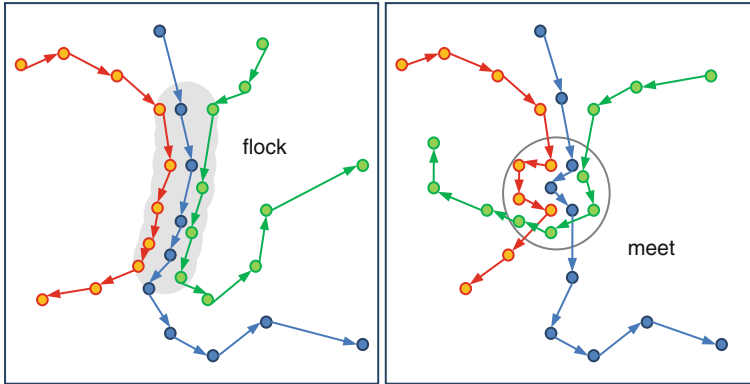


Fig. 17.14 Examples of flock (flocking behavior, *left*) and meet (meeting behavior, *right*) patterns of three trajectories

Combining trajectories with event data is useful because paths alone do not necessarily tell us why players navigated in a particular way, or if e.g. a group of players travelled together – something that is important to evaluate in any game where a strategic behavior includes a team of people supporting each other, for example first-person shooters and MMORPGs. Usually cooperative behavior is limited to a specific period of time, which can make spatiotemporal data analysis interesting.

To cluster trajectories, it is necessary to agree on a measure of similarity. Such a similarity measure might consider some aspects of the input and neglect other information. For example, we can distinguish between spatial and temporal similarity. Comparing trajectories based on spatial information might only compare the sequence of way points, whereas comparisons based on spatiotemporal information additionally considers that the way points were also visited at similar points in time. Finally, trajectory clustering also need to consider how to handle the dissimilarity between trajectories – it is very unlikely that two players move together for the complete period of playing a game. Common similarity measures for trajectory data are *closest pair distance*, *longest common subsequence*, *sum of pairs distance* and *dynamic time warping* (Zheng and Zhou 2011).

Trajectory analysis has established types of patterns which are indicative of specific trajectory behaviors. For example tracks, flocks and leadership patterns (Fig. 17.14 describes both types of pattern). A *flock* describes a set of at least n objects travelling together while staying in a disc with radius r for k consecutive points in time. A *meet* specializes the definition of a *flock* by demanding that at least n objects stay in a stationary disc of radius r for k consecutive points in time. While *flocks* might describe a temporary joined movement, a *meet* describes a spot where several objects are gathering. A final spatial pattern that can be analyzed based on relative motion patterns is the leadership pattern. In this setting, objects might join the movement of a leading object at some point of time. An extended description of these patterns and how they can be derived is given in Zheng and Zhou (2011).

Applying data mining techniques for trajectory data has been already employed for various use cases in game analysis (Thurau et al. 2004; Miller and Crowcroft 2009; Pao et al. 2010). In Miller and Crowcroft (2009), the authors examined character movements in *World of Warcraft*. In particular, the analysis is concerned with character movement in the Arathi Basin player-vs-player battle ground. In this closed setting, 15 players of two fractions compete over the dominance of five bases on an instanced map which separated from the open environment of the general game. Winning the game is strongly connected to capturing and holding the majority of basis for as long as possible. Thus, looking at the distribution of players and their movement on the map is assumed to reveal individual strategies as well as successful group strategies. Methodically, the authors examine three types of information:

1. The first is waypoint-based movement models. In this type of analysis trajectories can be shortened to the part connecting two waypoints. The way points may either be selected by domain knowledge (e.g. knowledge of the design of a game) or generated by spatial clustering (i.e. by analyzing where trajectories intersect). The idea with way point movement is that players travel between a small set of waypoints and use efficient routes on their travel. In the Arathi Basin data, waypoints could successfully describe the movement of so-called “patrolling” players which try to either capture or hold nearby bases.
2. A second type of information is spatial hot spots. A hot spot is an area of the map where characters spend most of the time. To derive hotspots the authors partition the map into a grid and count the seconds the characters spend in each grid cell. The cells with the highest counts are selected as hotspots. As expected the bases and the spawning points were selected as hot spots. The analysis of hot spots revealed a second type of players, termed “guards”. A guard stays close to a base for the complete game to defend it against turn over.
3. Finally, the authors examine flocking and grouping of characters. Methodically, they recorded characters moving together within a 30 yards diameter for several seconds. Interestingly, this analysis did not reveal typical group movement which can be derived to the fact that players might die and respawn at the nearest graveyard. Thus, during the game most of the movement will be spent to rejoin other players.

One of the most examined tasks in game analysis is detecting bot programs controlling an avatar in a MMORPG instead of a human player. Since movement is in many games the most common action of avatars and units, it is possible to distinguish bot controlled from human players by analyzing their movement in the virtual environment.

In Pao et al. (2010), the authors propose to detect bots by analyzing trajectory information in the FPS *Quake II*. The authors propose two representations to describe the characteristics of player movement. The first is the distribution of the step size. The step size corresponds to the distance a character moves between two points of time. Thus, if a character waits the step size is 0 and the faster the character is moving the larger is the step size. For a trajectory covering n steps, we can now obtain a distribution over the step sizes indicating how many percent of the steps the

character made had a certain step size. To represent this distribution a histogram is used where each bin represents a certain distance interval. To compare a pair of these histograms the authors propose to employ the standard measure for comparing distributions, the KL divergence (Kullback and Leibler 1951). A second approach measures the change in step size and direction relative to the previous point of time. Instead of histograms, the changes in both features are described by a normal distribution. Based the distribution of step size and directional change, it is possible to compute a log likelihood for a trajectory based on the Markov assumption. The similarity of two trajectories is based on three of these Markov models. One for each trajectory on its own and a joined model being generated from the concatenation of both comparison partners. If the joined model generates both trajectories with the same likelihood as the individual models, the trajectories are considered to be similar because it is assumed that they were drawn from the same distribution. To determine whether a trajectory belongs to a bot, distance-based classification methods, in this case instance-based learning and support vector machines (Han and Kamber 2001) are applied. Furthermore, the authors propose to use feature reduction techniques like ISOMAP to further improve their result. The results achieved more than 96% of prediction accuracy for detecting three types of common *Quake II* bots based on trajectories corresponding to 100 s.

Another paper analyzing the movement behavior of human players is Thureau et al. (2003). However, the purpose in this research lies with the construction of more realistic bots (autonomous entities, e.g. NPCs). The work is rather interesting because it demonstrates that bot detection and bot construction are just two sides of the same coin. Though building a statistical process describing human behavior can help to detect bots which do not fit to the model, building a bot behaving according to human behavior is more effective with respect to evading detection. Thus, in order to detect modern bots, bot detection methods have to consider that professional game bots already use statistical models to appear more human. Thus, bot detection needs to rely on similar techniques as well. However, if gaming companies employ the same technical know how as the bot providers, they usually have two important advantages: (1) The first is that making a bot as human as possible usually decreases the advantages of using bot in the first place. (2) The second advantage is the availability of data. Though a bot programmer might have the technical knowledge to construct sophisticated statistical models, the availability of data for training the models is usually strongly limited.

To model human like movement, the method proposed by Thureau et al. (2003) goes through multiple steps. The first is to build a graph model to describe the actual locations of players and the movement between them (a basic example is shown in Fig. 17.15). Thus, the graph does not contain any paths leading through dangerously open territory because real players would not move this way. To model movement behavior the algorithm therefore needs to select targets where a player would move in a certain situation. In this work, the situations are clustered based on the characteristics of the players, i.e. health, ammunition, weapons or armor. For example, a character running low on armor might aim for a well known spawning point to collect a new one. Using clustering allows us to detect typical situations which would

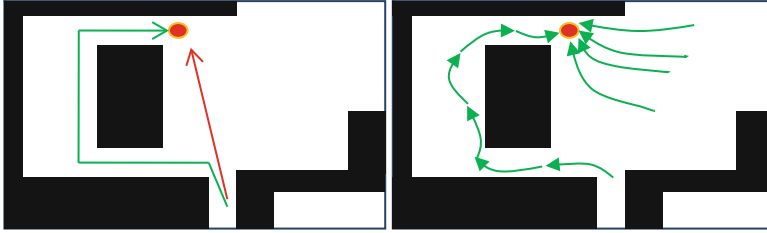


Fig. 17.15 The *left* figure shows an observed movement path, where the green arrows denotes the actual observation, the red one the shortest path to the goal. The *right* figure shows the learned potential field forces, which would guide the agent along the longer path, when approaching the goal from the initially observed direction

be followed by typical movements along the derived trajectories. The attractiveness of the locations under the current typical situation is modeled by a potential field. The idea behind this model is rather to always go in directions offering the most advantage in contrast to its distance on the movement graph. Though this method is mainly concerned with generating good trajectories for computer controlled entities, it also relies on a model of player behavior by distinguishing different situation which would lead to different actions. In the next section, we will turn to more compound descriptions of human behavior.

17.4.6 Behavioral Analytics

Other information that can be derived in association with the logging of spatial trajectories in games includes the type of activity a moving object (e.g. a player) is currently undertaking. In real-life geo-information systems contexts, a lot of the necessity for deriving this type of information is based on the incompleteness of the available data. For example, a classical task in this area is transportation mode detection, i.e. to predict whether a cellular owner travels by bus, bike or car. When analyzing behavioral telemetry data from games, there is usually much more information available as compared to real-life situations and therefore, determining the state a game entity is currently in is not necessary. However, the analysis methods for modeling behavior in GIS applications are very sophisticated and thus, methods for activity recognition can be used to mine complex behavioral patterns as well. For gaming applications, activity recognition allows for combining spatio-temporal movement data with activity patterns.

Examples for behavioral patterns in a computer game might be so-called build orders in a strategy game like *Starcraft* (Weber and Mateas 2009). A build order describes the order of buildings and units that has to pursuit a certain strategy (see e.g. http://wiki.teamliquid.net/starcraft/Category:Zerg_Build_Orders). For example, a player wanting to start an early surprise attack on his opponent will select a strategy favoring the training of attack units as early as possible instead of collecting

a lot of resources first. Finding out about starting strategies yields valuable information about game balance. Since the opening moves in most games tend to follow a rather static pattern, most players can follow them easily. Thus, finding a starting strategy without a proper counter indicates a flaw in game balance.

Another example application for activity recognition is account kidnapping in MMORPGs. While not an example of spatial analytics, it is an interesting example of where activity recognition can be applied in games: In this case a hacker acquires an account password and then logs into the account with the purpose of stripping the characters connected to the account from all of their valuables and distribute them to other accounts for selling the unrightfully acquired wealth to customers paying with real-world money. Though the first line of defense is usually to increase the account security, recent events show that account kidnapping is still at its height. To limit the damage done and the effort for restoring the customer account, detecting the hacker while in the process of stripping the account offers many advantages. A solution to this problem could be to learn the typical behavior of hackers and automatically lock the account in case the learned behavior is observed. An example behavior could be described by multiple paths between the bank, the auction house and the post box connected with selling events and the systematic change between all characters being connected to the account.

Though the approaches and applications for behavioral analytics are quite heterogeneous, there are two things all methods have in common:

1. The first is that *behavior is considered as a successive sequence of actions*. Thus, behavior is considered a set of several single actions that are done to achieve a certain goal. The sequence of actions might play an important part but in general derivations from the exact order might also lead to satisfactory results.
2. The second important aspect to consider is the *context of the behavior*. Thus, different situations will lead to different types of behavior. An example is the model for modeling human-like player behavior described above (Thureau et al. 2003). By clustering the player status and connecting it to the subsequent behavior, the model distinguishes first the context and then, describes the movement. For example, the behavior of a player running out of ammo will be to collect new one, while the behavior of a player recognizing that his opponent has no ammo left will be to hunt him down while he is more or less defenseless.

Methodically, the tool box for describing behavioral patterns contains simply methods for sequential data (Han and Kamber 2006) such as sequential pattern mining and Markov Chains. Both methods are applicable to cases where all the actions describing the behavior can be directly observed. If these steps are not directly accessible, methods like hidden Markov models, latent variable analysis or conditional random fields may be applied (Zheng and Zhou 2011). In these cases, it is only assumed to make observations being connected to the actual behavior. Though, it might appear that most applications in game analytics are based on clearly observable actions, there are as well use cases requiring the more complicated observation based models. One reason for the use of models that distinguishes between the

observed actions and the intended ones is fault tolerance. A player showing a complex action pattern might tend to make mistakes. Thus, the observed action might not be the intended one.

Furthermore, observation based models are often more tolerant with variations of less important parts of a behavior. Finally, in some games some player actions cannot be observed because they are not done in the game. An example is poker bots calculating win probabilities offline. Furthermore, team communication in computer gaming is often done by external voice chats. A final technical reason may also be that storing all actions in an online game for later analysis might cost too many resources. Thus, the game analytics experts has to suffice with information being currently collected, even though it may be possible to extract and store more detailed information.

There are already several papers describing player behavior as sequential patterns. Let us note that trajectories being described by relative motion patterns also fall into this category. Breining et al. (2011) applies sequential pattern mining to modeling player behavior. The approach describes player behavior as a string or sequence of the performed player actions. In addition to this basic approach, several extensions to the plain sequential model are introduced. A first extension is considering a time step of each action allowing for distinguishing how fast a certain action sequence was performed. A second extension is adding a spatial position to each action. Thus, it is possible to model trajectories and see whether there is a certain unknown correlation between the actions and locations on the map. Finally, an action sequence could consider the context of an action like the actual state of the avatar. This last information is especially useful to determine changes in the strategy. After the description of player behavior the method employs a suffix tree to derive frequently observed action sequences for different contextual information within the game. The experiments were performed on *Quake III* data and showed successful results for player identification and finding local behavior patterns.

A system modeling player behavior as sequential patterns is SABAF (Sequence Alignment Analysis of Player Behavior) (Shim et al. 2009). The task of SABAF is to predict whether players will be inactive for at least 30 days for the MMORPG Everquest II. To describe the player activity the method describes players by a sequence of 25 actions, e.g. *finished quest*, *killed a monster* or *was killed*. Based on this description players are compared with local and global sequence alignment. To make a prediction SABAF determines the most similar players in sample database containing active and inactive players. Based on the most similar cases in the sample database the system decides whether it is likely that the player will return to play the game or just quits. The results indicate that the order in which the events occurred seems to be important to the behavior of the player.

In Matsumoto and Thawonmas (2004) the authors apply hidden Markov models on player classification. Though the method is not performed on real game data but on a dedicated simulation environment the approach is very interesting because it distinguishes between observations and actual activities. In the paper, the observations are basic actions a game entity might undertake, such as walk, attack, chat or

picking up various items. The hidden states represent activities such as hunt, fight or go for power-up. An example, where the usefulness of this approach might become apparent is the observation that a character walks around the virtual environment which could indicate a lot of activities, like hunting for items, fleeing from an enemy or maybe the character is just bored. In combination with other events the actual activity might become much more apparent. For example, a lot of pick up events in combination with walking would indicate the hunt activity. To conclude, using a model including latent or hidden states allows for modeling the intentions behind the actual actions that can be observed.

The methods being discussed so far describe the behavior of game entities in an isolated view, i.e. the behavior is categorized and judged only with respect to the intentions and the actual actions of the player. However, a large part of playing is strongly dependent on the behavior of other players. To judge whether a strategy works out or not, we have to consider what the other players are doing. For example, choosing a proper opening in a real-time strategy game is strongly dependent on what opponents and team mates are doing. Mining collaborative and antagonistic behavior is a very complex challenge without a lot of general solutions so far. The general problem is to decide which actions should be considered as related and which are not. By treating everything as related to everything else, we usually run into an over-fitting view of the world displaying nothing but unique events instead of universal behavioral patterns. On the other hand, the key component of team play might only be described by complex spatial relationships.

There are first solutions to mining team play in the area of analyzing real sports matches. For example, Kim et al. (2011) propose methods for spatiotemporal analysis of soccer. Besides a general model to describe the collective movement of the players and the ball in a soccer match, the authors propose to employ certain morphological descriptors to model different constellations of the four defensive players. The morphological measures rely on width and depth of the complete defensive formations. Furthermore, the authors derive distance and angle vectors to measure the relative positions of the players in the formation. Having the group formation at a certain point of time it is possible to detect the change in certain situations. For designing sports games, this type of analysis is very important to model realistic team behavior.

17.5 Conclusions and the Way Forward

In this chapter the current directions in spatial game analytics and –visualization across industry and academia has been described, reviewed and categorized, and a series of case studies presented to provide concrete examples of method application and recommendations for use. There is a substantial amount of relevant research available that ties in with these topics which there has not been space to cover here, e.g. advances in adaptive gaming and AI. Some of these are discussed in other chapters in this book, e.g. Chaps. 6, 7, 9, 12, 14, 18 and 19.

Spatial game analytics can be advantageous in comparison with traditional non-spatial analysis of player-derived telemetry because it does not reduce the dimensions in the data, but deal with the perspective that the games are experienced through. It also allows analyses which are not possible in the non-spatial domain, e.g. trajectory analysis, which is useful in determining asset use. Furthermore, spatial game telemetry is characterized by being of a very high quality as compared to real world with no or less measurement errors, which alleviates some of the main challenges with performing spatiotemporal analysis. Existing case studies indicate that spatiotemporal data from computer games can be complex, which indicates that selecting simple data representation models can lead to the most general models. When fielding new techniques on game data, it is therefore advisable to start simple. There are decades of research in real-world spatiotemporal analytics available which can be adapted to game analytics, and established data mining methods are widely applicable. However, the key of successful knowledge discovery is to design a complete process. Starting with clear and useful targets and ending with clear evaluation metrics. Furthermore, improving data quality is in most cases much more important than selecting and tuning the data mining algorithms.

Capturing spatial behavior from players can be bandwidth heavy. For example, tracking trajectories using 1-s frequencies results in a substantial number of data points, that need to be stored locally client-side or transmitted to a server-side collection layer. Multiply this with a few thousand for an MMOG, and there could be trouble. There is no easy solution to the bandwidth bottleneck, and usually some sort of sampling regime will need to be instigated to alleviate problems, depending on the specific game type. There is a trade-off in precision between tracking positions with a high vs. a low frequency, e.g. every second vs. every minute, and which approach to use depends on the requirements of the specific stakeholders. More intelligent tracking regimes, where e.g. more detail is capture from players in MMOG instances than when they are operating in a shared game environment, also form venues for the conflict between bandwidth (and database space/processing power) and sufficient detail and depth of information to enable analytics. During production and testing, usually there is opportunity to measure everything analysts could desire, due to the large degrees of bandwidth available.

Spatial game analytics is in its infancy, and there is a substantial number of challenges and open questions, notably the lack of a general view on which solutions to use in which cases: The reliance on analytics tools from sectors outside of game development means that the barrier of entry for non-experts can be relatively high; however, the current innovation in game analytics is probably going to change this in the coming years as more and more tools become available. Current knowledge is fragmented, released across a variety of research fields and industrial sectors, and generally case-based. Finally, methods for describing and mining collaborative and antagonistic behavior are still in development – spatial game analytics could become the “killer application” for this direction. Spatial game analytics – and game analytics in general – have a potential for transforming game development in much the same way it has done in sectors such as marketing, web applications and environmental modeling, however, games are unique beasts and it remains to be seen how big an influence analytics will have.

About the Author

Anders Drachen, Ph.D. is a veteran Data Scientist, currently operating as Lead Game Analyst for Game Analytics (www.gameanalytics.com). He is also affiliated with the PLAIT Lab at Northeastern University (USA) and Aalborg University (Denmark) as an Associate Professor, and sometimes takes on independent consulting jobs. His work in the game industry as well as in data and game science is focused on game analytics, business intelligence for games, game data mining, game user experience, industry economics, business development and game user research. His research and professional work is carried out in collaboration with companies spanning the industry, from big publishers to indies. He writes about analytics for game development on blog.gameanalytics.com, and about game- and data science in general on www.andersdrachen.wordpress.com. His writings can also be found on the pages of Game Developer Magazine and Gamasutra.com.

Matthias Schubert, Ph.D. is an Assistant Professor at the Institute for Informatics at the Ludwig-Maximilians-Universität at München, Germany. He finished his PhD thesis on advanced data mining techniques for compound objects in 2004. His research interests are in spatial and multimedia database systems, similarity search, knowledge discovery and data mining. In particular his recent research is in spatial query processing in transportation networks, medical imaging and game analytics. He has published over 40 refereed conference and journal papers.

Suggested and Broadly Accessible Readings

- Dankoff, J. (2011, September 12). *Game telemetry with playtest DNA on assassin's creed*. The engine room. URL: <http://engineroom.ubi.com/game-telemetry-with-playtest-dna-on-assassins-creed/>
- Demers, N. (2008). *Fundamentals of geographical information systems*. URL: <http://www.amazon.com/Fundamentals-Geographical-Information-Systems-Michael/dp/0470129069>
- Drachen, A., & Canossa, A. (2011). *Evaluating motion: Spatial user behaviour in virtual environments*. URL: http://andersdrachen.files.wordpress.com/2011/01/05_drachen_ijart.pdf
- Hoobler, N., Humphreys, G., & Agrawala, M. (2004). *Visualizing competitive behaviors in multiuser virtual environments*. URL: <http://www.cs.virginia.edu/~gfx/pubs/lithium/>
- Houghton, S. (2011). *Balance and flow maps*. URL: <http://altdevblogaday.com/2011/06/01/balance-and-flow-maps-2/>
- Kennerly, K. (2003). *Better game design through data mining*. URL: http://www.gamasutra.com/view/feature/2816/better_game_design_through_data.php
- Pruett, C. (2010). *Hot failure: Tuning gameplay with simple player metrics*. URL: http://www.gamasutra.com/view/feature/6155/hot_failure_tuning_gameplay_with_php?print=1
- Thompson, C. (2007). *Halo 3: How Microsoft labs invented a new science of play*. URL: http://www.wired.com/gaming/virtualworlds/magazine/15-09/ff_halo
- Zoeller, G. (2011). *MMO rapid content iteration*. URL: <http://gdc.gulbsoft.org/>

References

- Breining, S., Kriegel, H.-P., Züfle, A., & Schubert, M. (2011). *Action sequence mining. ECML/PKDD workshop on machine learning and data mining in games*. Athens/Greece.
- Bungie (2011). Source: <http://www.bungie.net/Online/HeatMaps.aspx>.
- Canossa, A., Drachen, A., & Sørensen, J. R. M. (2011). Arrgghh!!! – Blending quantitative and qualitative methods to detect player frustration. In *Proceedings of foundations of digital games conference*. Bordeaux: ACM Publishers.
- Dankoff, J. (2011, September 12). *Game telemetry with playtest DNA on assassin's creed*. The engine room. URL: <http://engineroom.ubi.com/game-telemetry-with-playtest-dna-on-assassins-creed/>
- Dankoff, J. (2012, May 24). *Game telemetry with playtest DNA on assassin's creed, part 3*. The engine room. URL: <http://engineroom.ubi.com/game-telemetry-with-playtest-dna-on-assassins-creed-part-3/>
- Demers, M. N. (2008). *Fundamentals of geographical information systems*. Hoboken: Wiley.
- DeRosa, P. (2007, August 7). Tracking player feedback to improve game design. *Gamasutra*. URL: http://www.gamasutra.com/view/feature/1546/tracking_player_feedback_to_php
- Drachen, A., & Canossa, A. (2009). Analyzing spatial user behavior in computer games using geographic information systems. *13th MindTrek*. Tampere: ACM-SIGCHI Publishers.
- Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behaviour in virtual environments. *International Journal of Arts and Technology*, 4(3), 294–314.
- Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player modeling using self-organization in tomb Raider: Underworld. In *Proceedings of the IEEE symposium on computational intelligence and games* (pp. 1–8), Milan.
- Ester, M., Frommelt, A., Kriegel, H.-P., & Sander J. (1998). Algorithms for characterization and trend detection in spatial databases In *Proceedings of the 4th ACM international conference on Knowledge Discovery and Data Mining (KDD)*, New York City, NY.
- Gagné, A. (2011). *Visually exploring player strategies with pathways, a visual analytics tool*. MSc. thesis, Simon Fraser University, Burnaby.
- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2012). Analysis of telemetry data from a real-time strategy game: A case study. *ACM Computers in Entertainment*, 10(3), Article No. 2.
- Ghanem, M., Guo, Y., Hassard, J., Osmond, M., & Richards, M. (2004). Sensor grids for air pollution monitoring. In *Proceedings of the 3rd UK e-science all hands meeting*, Nottingham.
- Han, J., & Kamber, M. (2006). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann Publishers.
- Han, J., Kamber, M., & Tung, A. (2001). Spatial clustering methods in data mining: A survey. In H. Miller & J. Han (Eds.), *Geographic data mining and knowledge discovery*. London: Taylor and Francis.
- Hawkins, D. (1988). *Identification of outliers*. London: Chapman and Hall.
- Hoobler, N., Humphreys, G., & Agrawala, M. (2004). Visualizing competitive behaviors in multiuser virtual environments. *IEEE visualization conference*, Austin.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco: Morgan Kaufman Publishers.
- Kim, J. H., Gunn, D. V., Suhuh, E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). *Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems*. Florence: Computer–Human Interaction (CHI).
- Kim, H.-C., Kwon, O., & Li, K.-J. (2011). Spatial and spatiotemporal analysis of soccer. In *Proceedings of 119th ACM SIGSPATIAL international conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, San Jose.
- Koperski, K., & Han, J. (1995). Discovery of spatial association rules in geographic information databases. In *Proceedings of fourth international symposium on large spatial databases* (pp. 47–66). Maine.

- Kriegel, H. P., Schubert, M., & Züfle, A. (2011). Managing and mining spatio-temporal data in massive multiplayer online games. In *Proceedings of the 12th international symposium on spatial and temporal databases (SSTD 11)*, Minneapolis, MN.
- Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1), 79–86. doi:10.1214/aoms/1177729694.
- Lewis-Ewans, B. (2012, April 24). Finding out what they think: A rough primer to user research, parts 1 and 2. *Gamasutra*. URL: http://www.gamasutra.com/view/feature/169069/finding_out_what_they_think_a_php
- Li, S. Z. (1995). *Markov random field modeling in computer vision*. New York: Springer.
- Longley, P., Goodchild, M. F., Macquire, D., & Rhind, D. (2005). *Geographic information systems and science*. Chichester: Wiley.
- Lu, C. T., Chen, D., & Kou, Y. (2003). Algorithms for spatial outlier detection. In *Proceeding of 3rd IEEE International Conference on Data Mining (ICDM 2003)*, Melbourne.
- Matsumoto, Y., & Thawonmas, R. (2004). MMOG player classification using hidden Markov models. In *Proceedings of 3rd International Conference/Workshop on Entertainment Computing (ICEC)*. Eindhoven: The Netherlands.
- Medler, B., & Magerko, B. (2011). Analytics of play: Using information visualization and gameplay practices for visualizing video game data. *Parsons Journal for Information Mapping*, 3(1).
- Medler, B., John, M., & Lane, J. (2011). Data cracker: Developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of the 2011 annual conference on human factors in computing systems* (pp. 2365–2374). Vancouver, BC, Canada.
- Mellon, L. (2009). *Applying metrics driven development to MMO costs and risks*. Versant Corporation.
- Miller, J. L., & Crowcroft, J. (2009). Avatar movement in world of warcraft battlegrounds. In *Proceedings of netgames 2009*. Paris: IEEE Publishers, ISBN: 978-1-4244-5605
- Moura, D., Seif El-Nasr, M., Shaw, C. D. (2011). Visualizing and understanding players' behavior in video games: Discovering patterns and supporting aggregation and comparison. In *Proceedings of the 2011 ACM SIGGRAPH symposium on video games (Sandbox '11)* (pp. 11–15), ACM: New York. ISBN:978-1-4503-0775-8, doi:10.1145/2018556.2018559
- Pao, H.-K., Chen, K.-T., & Chang, H.-C. (2010, September). Game bot detection via Avatar trajectory analysis. *IEEE Transactions on computational intelligence and AI in games*, 2(3), 162–175. doi:10.1109/TCIAIG.2010.2072506. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5560779&isnumber=5571836>
- Pruett, C. (2010). Hot failure: Tuning gameplay with simple player metrics. *Game Developer Magazine*, September 2010. Available from: http://www.gamasutra.com/view/feature/6155/hot_failure_tuning_gameplay_with_php?print=1
- Ramero, R. 2008. *Successful Instrumentation*. Presentation at the Game Developers Conference. San Francisco, CA.
- Shekhar, S., & Huang, Y. (2001). Co-location rules mining: A summary of results. In *Proceedings of the 7th international symposium on spatial and temporal databases*. Redondo Beach, CA, USA: Springer Publishers.
- Shekhar, S., Schrater, P. R., Vatsavai, R. R., Wu, W., & Chawla, S. (2002). Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transaction on Multimedia*, 4(2), 174–188.
- Shekhar, S., Lu, C., & Zhang, P. (2003). A unified approach to detecting spatial outliers. *GeoInformatica*, 7(2), 139–166.
- Shim, K. J., & Srivastava, J. (2009). Sequence alignment based analysis of player behavior in massively multiplayer online role-playing games. In *Proceedings of the IEEE ICDM-10 workshop on domain-driven data mining*.
- Thawonmas, R., Kashifuji, Y., & Chen, K. T. (2008). Detection of MMORPG bots based on behavior analysis. In *Proceedings of ACM SIGCHI international conference on advances in entertainment technology 2008 (ACE 2008)* (pp. 91–94). Yokohama.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine*. No. 15.09. Available at: http://www.wired.com/gaming/virtualworlds/magazine/15-09/ff_halo

- Thureau, C., Bauckhage, C., & Sagerer, G. (2003). Combining self organizing maps and multi-layer perceptrons to learn bot-behavior for a commercial game. In *Proceedings of the GAME-ON conference* (pp. 119–123). London: United Kingdom.
- Thureau, C., Bauckhage, C., & Sagerer, G. (2004). Learning human-like movement behavior for computer games. In *Proceedings of 8th international conference simulation of adaptive behavior* (pp. 315–323). MIT Press.
- Weber, B., & Mateas, M. (2009). A data mining approach to strategy prediction. In *IEEE symposium on computational intelligence in games* (pp. 140–147). Milan.
- Zheng, Y., & Zhou, X. (2011). *Computing with spatial trajectories*. New York: Springer.
- Zoeller, G. (2010). Game development telemetry. *Presentation at the game developers conference 2011*. San Francisco, CA.
- Zoeller, G. (2011). MMO rapid content iteration. *Presentation at GDC online 2011*. Game Developers Conference. San Francisco: CA. URL: <http://gdc.gulbsoft.org/>.

Chapter 18

Visual Game Analytics

Ben Medler

Take Away Points:

1. Visualizing and analyzing game data means taking part in a process of collecting, manipulating, representing, examining and disseminating that data.
2. There are already a diverse set of visual game analytic tools being used by developers, third-party services and players.
3. A diverse set of tools to choose from means game data analysts have to decide which analytic solution is best for them.
4. Choosing to build a visual game analytic tool for game developers, for example, means working with the developers to create a tool that can answer the questions they have about their players, as the case study in this chapter illustrates.

18.1 Introduction

Checking my map I notice I am close to the target. I continue to dig, forging ahead while laying torches to furnish a small amount of light for me to see. I check my map again, there should be an opening ahead. Then I hit it, a wide, dark chasm stretching far below me. I light a torch and move in to attack the skeleton inhabitants. Once the boney menaces are dispatched I am able to acquire my prizes: a couple pots, a treasure chest and a vein of gold ore. I quickly collect everything in reach, check my map and continue downwards.

From this depiction you can assume I am playing a game, one complete with a map, skeletons and treasure. And you would be right, except for the map part.

B. Medler, Ph.D. (✉)
Chief Creative Office, Electronic Arts, Redwood City, CA, USA
e-mail: benmedler@gmail.com

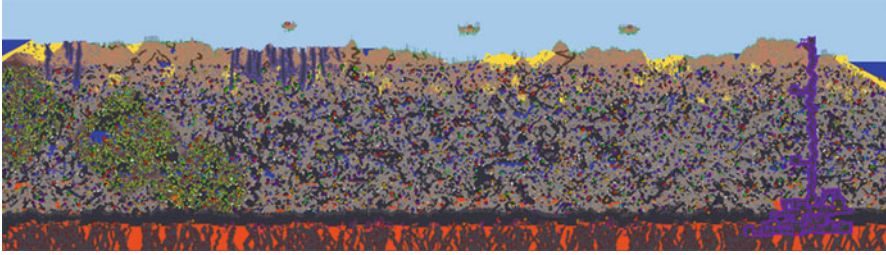


Fig. 18.1 Terraria players have built mapping tools, which visualize their Terraria game maps, a feature not available in the game (Used with kind permission of Mike M. (alias: Noroom))

The map I refer to is not part of the game. It is instead a visualization of the game world rendered from one of my saved games, rendered using a program built not by the game’s developer but a player. The program is called *MoreTerra* (MoreTerra 2011); it was built to be used with the game *Terraria* (Re-logic, 2011). Loading a *Terraria* saved game into *MoreTerra* produces an image similar to the one found in Fig. 18.1. *Terraria* is played in a two dimensional world, which is rather large compared to the player’s character, and the gameplay consists of the player exploring the world using various tools to literally dig into the world terrain. Players collect resources, fight monsters and craft new items as they explore, causing *Terraria* to act as a large sandbox or terrarium. Players are not given a map in *Terraria*, however, and therefore *MoreTerra* was created. With *MoreTerra* players can find resources, treasure and even floating islands existing in their world. It is basically a high powered surveying tool, the type that most real world companies that gather natural resources wish they had.

MoreTerra offers *Terraria* players a different perspective on their gameplay. One can argue that giving players a complete map defeats the purpose of a game built around exploration. But I argue that a map, such as the one *MoreTerra* offers, can engage players in other ways, giving them identifiable goals to pursue (such as collecting a specific resource), and provides a visual record of their world they can share (in a way, giving them the chance to actually view their world as a terrarium). This is how *MoreTerra* alters a player’s perspective about their gameplay, through visualizing game data. Although, visualization does not have to stop at rendering a world’s configuration, it can further be used with a number of game-related datasets: player behavior events, sales figures, server performance values, etc. Combining visualization and game data, of any type, offers additional ways players and developers can approach their data from other perspectives, whether inside or outside a game experience.

While many chapters in this book discuss the importance of visualization of metrics data, only two chapters focus solely on visualization and visual analytics systems for games: this chapter and the following chapter (Chap. 19). In this chapter, I offer an overview of a domain known as visual game analytics, examining the link between game data and visualization. Combining the field of game analytics, one which “focuses on the systems and methods used to analyze game-related data (Medler et al. 2011),” with visualization techniques introduces visual game analytics – a field that offers methods

to collect, transform, manipulate, visualize, analyze and disseminate game-related data in the hopes of altering perceptions in regards to that data. I encapsulate visual game analytics by presenting the domain in three contexts, approaching the domain from different angles, including: elaborating on visualization theories, providing examples of current visual game analytic systems, and describing how an analytic tool can be deployed as part of a game's development process.

The first section is devoted to theories of visualization. Those interested in a brief theoretical background describing why visualization is important for analyzing data will benefit from this section. Games are a unique medium in relation to visualization, because games rely heavily on visual components. Presenting information in games often relies on those visual components and game developers can benefit from understanding visual perception theories and frameworks for creating visualizations for data analysis. Additionally, one major purpose of game analytics is to study game-related data outside of an actual game environment. Combining visualization theories with game analytics allows for the development of visual analytic tools used to analyze data over time, gaining insights about player behavior, hardware statistics and market performance.

The second section moves from the theoretical to the practical. Current visual game analytics tools are covered, detailing the groups who use the tools, what data is visualized and for what purpose. Readers looking for a snapshot of the current state-of-the-art in visual game analytics will find this section useful. Three major groups are discussed and play a role in shaping the scope of visual game analytics: game developers, third-party services and players. Game developers build and use analytic tools in order to gain insights from the trends and patterns they find in the game data they collect. Depending on the type of data collected from a game those insights span the gambit from understanding how players are behaving in a game to tracking software bugs. Next, third-party services often leverage analytic tools to disseminate aggregated or collected game-related data to their own community of users. Last, players are sometimes given analytic tools by a game developer or third-party services as an additional feature to use with a game. Some players, however, build their own analytic tools to augment their gameplay experience and to analyze their own data. Each group is presented in relation to the visual game analytic tools they use and how those tools aid each group in leveraging game data for their benefit.

The third and final section explores how a visual game analytic tool is built, readers wishing to develop their own tool may find this section helpful. Presented as a case study, I describe a research and development project at Electronic Arts where I helped build a visual game analytic tool for the game *Dead Space 2 (DS2)* (Visceral Games, 2011). The tool, known as Data Cracker, was built for game developers on the DS2 team and offered a way to help balance the multiplayer component in DS2. While the exact technical construction of the tool is not described in great detail, the overall insights our team discovered during the tool's development are laid out across three categories: production, functionality and communication. Production insights include the steps required to successfully deploy a visual game analytic tool including when development should begin and how the life cycle of the tool should be maintained. Functional insights point to specific functionalities the tool required

in order to aid developers in the data analysis process. Communication insights consist of the interactions my team had with the game developers and how relationships between the two groups affected the development of the tool.

Three sections constitute this chapter and explore why visual game analytic tools are needed, what tools are used and how these tools are developed. This chapter aids researchers who wish to further understand how visualization tools can alter game development or how these tools affect the overall game experience. For developers and players, this chapter provides a vital resource for understanding what visual game analytic tools exist and how they can create their own tools. Following this chapter, Chap. 19 will focus on visual analytics techniques for other game genres, including Real-Time Strategy (RTS) and Role-Playing Game (RPG), where the system development team integrated their system with other game companies.

18.2 Visualization Theories

Visualization is a concept tightly entangled with data and the abstract notion of “data” seems to be everywhere in our globally connected world. Data.gov (Data.gov 2010), a website created by the United States Government, provides “data generated across all federal agencies (Orszag 2009)” allowing the world to investigate the activities of the U.S. government through a mélange of datasets. Grassroot citizen movements, such as the one led by Arun Ganesh, an Indian designer, was able to crowd-source Chennai’s populous to gather data about the city’s local bus routes and create a comprehensible route map which had been officially “under construction” for 6 years (Ganesh 2010). From the largest government institutions to the loosely connected people that amalgamate spontaneously on the internet, gathering and applying data has become the norm.

Increased accessibility to data also comes at a price. Concepts like “information overload” (Edmunds and Morris 2000) and “information anxiety” (Wurman 2001) are said to hinder decision making and decrease efficiency due to receiving too much information, where information is often used synonymously with data. A few of the causes attributed to information overload in Eppler’s and Mengis’ review of the related literature include: uncertainty of information (the level of ambiguity, novelty, complexity, and intensity), complexity of tasks using the information, and technology advancement such as storage capacity or ease of access. Having too much data, even data that is deliberately collected, can be a burden to the receiver, causing information overload.

Shirky argues that the problem of having too much information is not information overload, however, but a filter failure (Shirky 2008). In the past media outlets – publishers, broadcasters, filmmakers – filtered media for quality. With the rise of digital media and participatory culture those filters have ceased to provide adequate quality filters. As Eppler’s and Mengis’ write:

Simpson and Prusak (1995) argue that modifying the quality of information can have great effects on the likelihood of information overload. Improving the quality (e.g., conciseness, consistency, comprehensibility, etc.) of information can improve the information-processing capacity of the individual, as he or she is able to use high-quality information more quickly and better than ill-structured, unclear information (2004, p. 331).

The inability to process large quantities of information is not caused by the abundance of data, but the quality of how that data is presented or accessed.

Manipulating the quality of data is the focus of many research fields, including data mining (Kantardzic 2003), information visualization (Card et al. 1999) and visual analytics (Thomas and Cook 2005). These fields have benefited over the last few decades from digital technologies even when contrasted against the argument that the rise of digital technology is also a cause of information overload (Schultze and Vandenbosch 1998). Those benefits are caused by the affordances of computers to act as external aids for users. As Norman writes, “without external aids, memory, thought, and reasoning are all constrained” and using external aids “enhance(s) our cognition (Norman 1994).” He argues that one type of external aid is cooperative social behavior, in which groups of people work towards common goals, sharing information amongst each other. The other type of external aid is “cognitive artifacts,” which Norman describes as tools that help externalize thoughts. Pencil and paper are cognitive artifacts allowing us to record information outside of our mind, enabled by our reading and writing skills (Norman 1994). Computers, or digital devices, act as external aids as well.

We extend our cognitive skills by using any type of external aids, digital or otherwise, and our most important sense that enhances external aids is visual perception. “Vision is not only the fastest and most nuanced sensory portal to the world, it is also the one most intimately connected with cognition (Few 2009).” Information visualization (Infovis), one of the fields previously mentioned for altering the quality of data, is built around the concept that vision is a powerful cognitive human perception and uses “computer-supported, interactive, visual representations of abstract data to amplify cognition (Card et al. 1999).” Infovis is “computer-supported” and directly related to digital technology, unlike concepts, such as information graphics, which are often static or printed visual representations of information (Tufte 1983). Using computer-supported systems to overlay data on a map automatically or present data using interactive charts, allowing users to drill down into a dataset, are examples of the power Infovis has over static graphs. Visualizing data using computation helps someone understand a dataset by acting as an external aid to their cognition, one that is dynamic and can be further explored.

Vision enhances external aids for data analysis whether it is augmented using computer-supported practices, such as Infovis or not. Ware notes (2000) that vision, in respect to analyzing data, has five key advantages as it allows:

- The ability to comprehend huge amounts of data.
- The perception of emergent properties that were not anticipated.
- The ability to find problems with the data itself.
- The understanding of both large-scale and small-scale features of the data.

The ability to form hypotheses.

Increasing the quality of data using visualization, Ware argues, increases a user’s ability to arrive at informed decisions based on the data. Each of Ware’s stated advantages are the basis for information visualization; visualizing data helps analysts overcome information overload, form hypotheses and identify pattern based on their data.

While Infovis focuses on visually representing data to aid cognition, other researchers have argued that a more comprehensive approach beyond Infovis is required. Instead of focusing on visual representation alone, an approach which includes the entire process of data gathering, visual representation, analysis and dissemination is required. This approach known as visual analytics is the “science of analytical reasoning facilitated by interactive visual interfaces (Thomas and Cook 2005)” and is an attempt to incorporate and extend Infovis practices (Keim et al. 2008). Visual analytics include areas such as (Thomas and Cook 2005):

- Data representation and transforming – Converting heterogeneous and dynamic datasets into a usable format.
- Analytic reasoning – Supporting assessment, planning, and decision making through analysis techniques and algorithms.
- Visual representations and interaction – Using Infovis and visual perception to explore data.
- Production, presentation and dissemination – Communicating analysis results to a variety of audiences.

This creates a workflow for analyzing data that spans the five areas discussed at the beginning of the chapter. Data is gathered, transformed, visually represented, analyzed and, finally, disseminated to external audiences. In contrast to Infovis, visual analytics combines Infovis techniques with accessory actions, data gathering and dissemination, that Infovis already performed, but was not focused directly upon.

Visualizing data helps analysts comprehend data and visual analytics particularly supports “assessment, planning, and decision making (Thomas and Cook 2005).” Although, this should not suggest that visual analytics is entirely based on utilitarian or quantitative methods of decision making. Pousman’s et al. (2007) research reports a growing trend in Infovis and visual analytics to provide a wide array of audiences with systems that do not solely focus on analytic reasoning. They call these types of systems “Casual Information Visualization” (Casual Infovis) which they define as: “The use of computer mediated tools to depict personally meaningful information in visual ways that support everyday users in both everyday work and non-work situations (Pousman et al. 2007, p. 1149).”

They separate Casual Infovis into three categories to correspond to the type of systems they found:

- Ambient Infovis: systems found in peripheral locations and provide abstract depictions of data.
- Social Infovis: systems which visualize social networks and allow users to interact with their social data.
- Artistic Infovis: systems with the goal of challenging preconceptions of data and representation.

Each of the three categories are example of how visualization tools are being used to create experiences that are not focused on the actionable, analytical decisions one can make by analyzing data. Instead, exploring data becomes a casual, enjoyable, even playful activity.

Casual Infovis, along with alternative visualization movements, such as Manovich's "Cultural Analytics" (2008) and Hall's "Critical Visualization" (2008), points to the diversity within visualization research – a diversity, which is beneficial when discussing how to combine games with information visualization and visual analytics. Games, too, have both a serious, business focused side while concurrently having a fun, playful side (Medler and Magerko 2011). Visual game analytics should reflect this diversity; by looking at the tools that are being used by different game-related audiences we can see how this diversity is already being represented.

18.3 Visual Game Analytics

Combining the theories of visualization with game analytics forms **visual game analytics**, a domain here defined as focusing on exploring game-related data visually. This is a sub-domain of **game analytics**, which does not need to be visual but includes purely numeric analysis (e.g. via data mining, Chap. 12). Here, "visualization of game analytics" is used interchangeably with "visual game analytics". It should also be mentioned that visual game analytics (visualization of game analytics data) should not be confused with **spatial game analytics**, which has many similarities but exclusively focuses on analyzing and visualizing game telemetry which includes a spatial component, e.g. player character trajectories (Chap. 17). The analysis of game-related data can take on many forms. In this chapter, I focus on the analysis of metrics derived from game telemetry, a type of 'game-related data,' which represents quantitative measurements gathered from gameplay, i.e. records of player behavior (see Chap. 2). Examples of game metrics include in-game events, such as a player's death or a player choosing which in-game items to use. However, many other forms of metrics exist outside of gameplay, including: sales of games (Chap. 4), biometric monitoring of the player's physiology (Nacke et al. 2009) and software metrics (Goodman 1993). Game analytics, as well as the game metrics which forms the input data in the analysis process, additionally can be argued to extend to include qualitative data, too. Attitudinal data (player opinions), surveys, videos, and player user-generated content are all examples of qualitative data which can be analyzed, turned into metrics or used to contextualize quantitative metrics.

While a wide variety of game-related datasets exist for game analysts to study, both in and out of games, there are also a number of audiences, which find game analytics useful. As discussed in Chaps. 3 and 4, as well as other chapters in this book, there are many stakeholders within the industry that benefit from game analytics, including:

- Customer Relation Manager – Social connections between players inside a game environment and handling customer support requests.
- Financial Manager – Game and micro-transaction sales.
- Game Designer – Recorded player gameplay associated with a game's mechanics and interactions among players.
- Producer – Production schedules.

- Programmer – Source code complexity and hardware performance.
- Network Engineer – Server performance and data archiving.

Each group requires certain datasets, which are collected before, during or after a game has been produced. Analyzing source code requires different data, and perhaps tools, compared to analyzing game sales, for instance. However, these audiences and datasets are never separated completely. Designers rely on programmers to create efficient code and financial managers rely on producers to keep a development schedule on time. While we can say that these jobs are different in their craft, the data they are working with is not always mutually exclusive to each group. For the purpose of discussion here, however, all of the game development disciplines will be combined together as a single developer group. Not, as just previously mentioned, due to their similarity but because there are additional groups, which use game analytics that need to be addressed. But note that different chapters of this book highlight different uses of game analytics, for example see Chap. 7 which investigates a telemetry system build for developers vs. Chap. 4 which primarily investigates end-user telemetry systems. In this chapter, I look at third-party services and players as two additional groups that have yet to be introduced.

Third-party services and players, also benefit from and propagate game analytics. Instead of using game analytics to produce better games, these two groups analyze game data to understand player behavior, connect players and, in the case of players actually playing a game, use game data to augment their gameplay experience. Third-party services achieve this by aggregating game data from various sources or collect their own data related to games. Aggregating and collecting data helps them create communities around not one game, but hundreds of games, which no single game developer has the means of achieving. Players are often indoctrinated into using game analytics, when game developers provide players with statistic tracking or player dossier websites that visualize each player's game data (Medler and Magerko 2011). Other times, players build their own tools for exploring game data, which may or may not be related to their personal gameplay data.

Each of the three groups – game developers, third party services and players – are discussed below in relation to the visual game analytics tools they use. Each group has their own goals when using game analytics and has decisions they need to make in order to reach those goals. Game developers and players share a common struggle of whether they should build or buy their analytic software, whether they should build their own or learn a ready-made tool. Third-party services often build their own tools but they must decide if they will aggregate data from other sources or attempt to collect their own game data for analysis.

18.3.1 Game Developers

Game developers, who wish to begin to capture and analyze data from their game, have to decide whether they want to build or buy their analytic tools to begin the process. Many off-the-shelf visual analytic tools can be used for analyzing a variety of

game related datasets if developers wish to buy their tools. Software such as Tableau ([Tableau Desktop](#)), Spotfire ([Spotfire](#)) and Mircrostrategy ([Microstrategy 1989](#)) are built to be general-purpose visualization tools that can be connected to multiple sources of data and servers. Other game developers, especially those who build games for the Flash platform, can also use online services, which provide analytic tools and the means for collecting game data as well. Mochibot ([Mochibot](#)), Nonoba ([Nonoba](#)) and Playtomic ([Playtomic](#)) are flash game metric services, out of about a dozen middleware providers, that also provide a variety of telemetry collection and visualization tools. Each service provides Flash game developers with an API, an application protocol interface, which allows the developer to send game data back to one of the three services for storage and later analysis. In the past, the type of data that was collected by services, like Mochibot, were simple, usually only gathering data related to whether the game was loaded on a webpage. These services have now expanded to collect data, such as player achievements, micro-transaction and other player gameplay events (such as score or deaths) (see also Chaps. 4, 6, 7, and 14).

Buying some of the standalone analytic tools, like Tableau or Spotfire, can be expensive for a game developer, with individual licenses costing thousands of dollars. Even in the case of services, like Mochibot, which can provide free service to smaller game developers, the developer must give up control over their data and risk having their data exposed to outside companies. Developers, thus, have the option of building their analytic tools. Instead of buying a tool, they pay their employees to create a tool, or use an open source solution, which can offer more control, but they must deal with the constant upkeep of the tool as well.

Microsoft's TRUE system (Kim et al. 2008) is an example of a visual game analytic tool built to aid game user researchers as they playtest games. When used in conjunction with play testing, the TRUE system records video feeds of each player testing the game, records game metric events and prompts the players with surveys to gauge their attitude during play. Video and attitudinal data contextualizes the captured game events providing user researchers with a greater understanding of why player testers were behaving in a certain way in the game. The TRUE system can also be externally deployed during beta tests, continuing to capture game events but losing the ability to capture video. Data captured after any test is visualized by the tool using various charts, graphs, heat maps, etc. If video was captured the tool synchronizes the event and attitudinal data with the video playback, referencing all three data streams seamlessly. The tool's visualizations are then used to create analysis reports or allow game designers to drill down into the data themselves to find relevant information.

A different tool built to aid programmers instead of user researchers is the Skynet system built by Zoeller (2010) at Bioware and discussed in Chap. 7 of this book. Figure 18.2 show a screenshots from the system. This system focuses on the game development process by monitoring developer behavior rather than players. Bug tracking, software metrics and social networking features are combined to create an online-portal where developers can both stay in touch with one another and have consistent access to their game's development status. While many features in SkyNet use spreadsheets to represent data, visual callouts, such as color-coding important information, are used, in addition to other visual graphs, to display information,

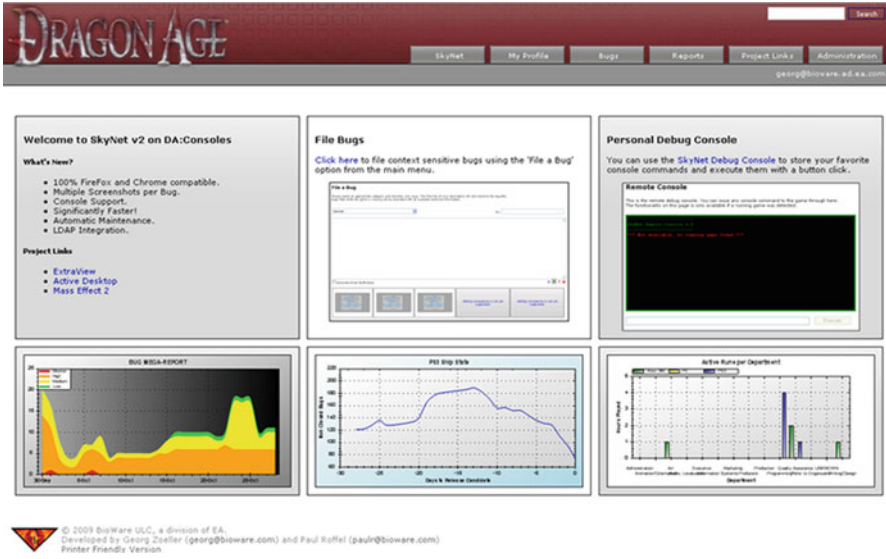


Fig. 18.2 SkyNet a visual game analytic tool built for monitoring game performance, bug tracking and developer behavior (© Bioware; courtesy of Bioware)

such as the number of bugs filed and fixed. Heat maps are also used to denote where in a game’s space the most crashes occur, even giving developers the option to click on the map in SkyNet and be transported to the location within the game environment. Whether developers decide to build or buy their analytic tools, visualization helps to analyze data and makes tools more accessible to a game development team no matter what type of game data they are analyzing.

18.3.2 Third Party Services

Third party services, unlike game developers, do not collect data from the games they make. Instead they focus on either aggregating data from other games or collect data from services they offer to players, like recommendation or social networking features. Third party services thus seek to bring together disparate games and datasets in order to leverage the capability to combine those datasets as a center piece for their service, often visualizing aggregated data in the process.

Aggregating data is used when a third party service, such as a game review website, can combine data from multiple different sources that have a similar data format. Metacritic (2001) is an example of a service that aggregates reviews about different media artifacts (Fig. 18.3). In particular, Metacritic aggregates game reviews and averages the reviews together to give each game an overall score, while also providing this information visually. Many game reviewing outlets use some



Fig. 18.3 Metacritic visualizes aggregated review scores for various media products including games (© Metacritic.com; courtesy of Metacritic.com)

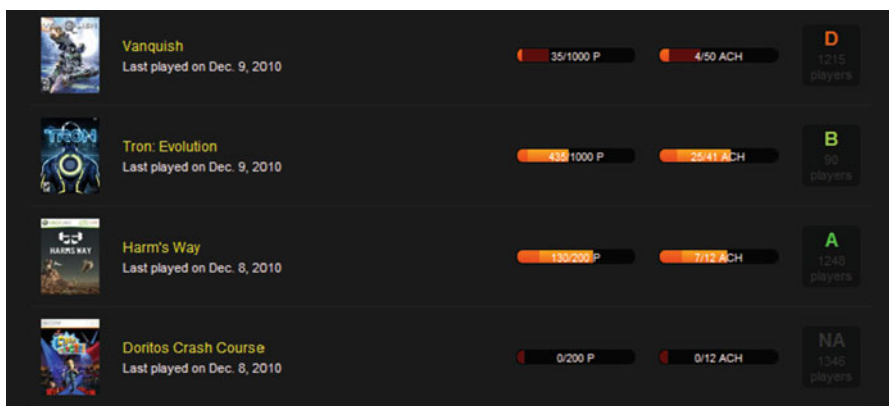


Fig. 18.4 Giant Bomb provides achievement visualizations for their users and compares each user's achievements with the whole Giant Bomb community (Image courtesy of GiantBomb.com)

form of a number system to rate games making it easy for Metacritic to combine these ratings together. If reviews do not a numeric score, Metacritic then will assign them a score. Metacritic takes each rating score, or assigned rating, and maps them onto a single zero to one hundred scale (0–100), thus unifying the aggregated data. These scores are subjected to weighted average as Metacritic curates the reviews, meaning not every added score is equally considered, but aggregated review scores is a valuable feature that Metacritic provides as a third party service.

Another example of a third party service that aggregates game data is Giant Bomb (Gerstmann and Davis 2008), shown in Fig. 18.4. A game news, wiki and community website, Giant Bomb generally relies on their community of game players to add to

the site's massive collection of game information. Players can add information about a game's mechanics, stories, characters, developers, etc. In this respect Giant Bomb actually acts as a third party system that collects data about games. Although, one specific feature of the website is related to aggregating data from different achievement systems that exist on a number of gaming platforms. Achievement systems are ubiquitous on most gaming platforms but do not transfer from one platform to another.

The service that Giant Bomb offers is a way to bring a player's achievement data from different platforms into one cohesive system. As a member of the Giant Bomb community, a player can connect their achievement data from four different sources: Playstation Network ([Sony](#)), Steam (Valve Corp. 2003), Xbox Live ([Microsoft](#)) or *World of Warcraft* (Blizzard Ent., 2004). Achievements from these sources are combined and visualized allowing a member to view all of their achievements in one place. Access to each member's achievements also gives Giant Bomb the power to determine additional information about achievements that would normally be impossible. For example, within the Giant Bomb community, the rarest achievements can be determined which can add additional value to those achievements. The average number of achievements earned per game can be found too, giving another piece of data that can be visualized to members for comparison with their personal achievement data. Combining different achievement datasets essentially allows Giant Bomb to create "a visual achievement catalog for their members to peruse achievements from a different perspective" (Medler and Magerko 2011).

Instead of combining data in the same way Metacritic and Giant Bomb do, other third party systems (or systems which span across many game developers) collect game-related data revolving around the act of playing a game. Steam (Valve Corp. 2003) a digital distribution and rights management service, collects data on players that use the service to buy and play games. Achievements are one of the standard data points collected by the service but Steam also collects data related to how much time a player spends playing games and each player's friend network. On Steam, how long a player plays a game is an indicator of how much a player enjoys playing that game, which aids Steam in recommending further games to that player. A player's friend network is used to show which friends are currently playing a game and can help make it easier for players to compete with their friends or join multiplayer matches together. Collected player data drives Steam's recommendations and social network features provide value to the service without having to actually collect much data from the games themselves.

GamerDNA (Radoff 2006) an online social network service for players, collects data in a similar manner to Steam except the service does not sell games. The service monitors its registered players as they play games on their PC/Mac or console platforms in order to understand each player's gaming habits, collecting behaviors such as when players are online, when they first start a game or earn achievements. Gathering this type of data allows GamerDNA to link together players, for instance, who recently started or continue to play a game, making it easier for those players to find other players in a single location, rather than having to use communication methods found in a single game or on a single platform. The service also has a number of surveys players can answer regarding what type of player they believe they are.

Visualization is used by third party services as a means to keep players focused on data that exists alongside games. Review scores, achievements, play times and friend networks are important to players but are not always provided to players by game developers. Third part services fill that role and use visual game analytics to aid players as they use player data to enhance their game experience.

18.3.3 *Players*

Players are in a unique position when it comes to visual game analytic tools. First, players are the audience that the other two game analytic users, game developers and third-party services, are attempting to engage with their games and services. Game developers and third-party services build quite elaborate visual game analytic tools for players to use in addition to games or other services like adding social networks. Second, players make games their own, projecting their own cultural habits, modifications and ideas on to their gameplay. Aiding this fact, some games allow players to create game mods or access game data through APIs which allow players to create their own visual game analytic tools (Maxis held a contest in 2008 involving creating online tools using the Spore API, for example (Maxis 2008)). Players, thus, have a choice, use the analytic tools developers and third-party services provide or make their own.

Beginning with the analytic tools that game developers provide, these tools are dossier systems, which track each player's individual performance in the game (Medler and Magerko 2011). Websites, such as Bungie.net (Bungie LLC 2004), offer the same level of meticulous player dossier record keeping when compared to professional sports (Medler 2009). Built for presenting data captured from Bungie's Halo game franchise, the website records data from four separate games (*Halo2*, *Halo 3*, *Halo 3:ODST* and *Halo: Reach* (Bungie LLC, 2004–2010)). Every kill made, point earned and objective completed is recorded during each battle, representing both single player and multiplayer gameplay. Player data is visualized using several visual forms including: line graphs for point progression over time, heat maps for detailing where events occurred in relation to the game map and sunburst graphs detailing when events occurred during separate sections of particular game. These visualizations are only offered within the online system and most player data cannot be accessed through the game. Limiting data access to Bungie.net creates a type of trophy room interface to each player's data. Awards and gameplay are put on display to explore outside of the game and far surpasses the investigative potential of systems relying solely on cumulative score keeping. (As of publication Bungie.net has ceased its player data tracking services and this service is now provided by Halo Waypoint.)

Some analytic systems are built right into the game and compare player data with their friends. Criterion's Autolog system (Electronic Arts Inc. 2010), built within the racing game *Need for Speed: Hot Pursuit* (Criterion Games, 2011), is a recommendation analytic system for promoting competition (Fig. 18.5). Autolog keeps track of a player's data related to their races (generally revolving around



Fig. 18.5 The Autolog system records gameplay data regarding race times in the game *Need For Speed Hot Pursuit* and recommends challenges for players to pursue (© Electronic Arts, Inc.; courtesy of Electronic Arts, Inc.)

their fastest race time) and compares their racing data with their friend's data. These comparisons are automatically turned into challenge recommendations, which are visually displayed within the game environment. Players are asked to beat the score or time of one of their friends receiving a reward for completing the recommendation. This creates an improvised playful environment amongst a community of friends where players choose which goals and competitions to pursue.


A different system, The Sims Exchange (Electronic Arts Inc. 2004) is one example where players can use past gameplay data to create their own stories related to their sims families they create in the game *The Sims 3* (Electronic Arts, 2009). The Sims Exchange is a service for sharing assets built for *The Sims* games as well as stories about player's personal play through of *The Sims*. Players take screenshots while playing the game and can create slideshow stories using the images they capture (Fig. 18.6). This is different from Bungie's system, which records data automatically, but in the similar manner a TheSims Exchange player is presenting a replay of their play experience with their personal interpretations of the events overlaid on top. It's a qualitative form of data analysis. Considering *The Sims 3* is supposed to represent creating stories using a doll house environment it makes sense that the players are given the ability to tell their stories by using The Sims Exchange.

Not all game developers are able to provide players with adequate means of analyzing game data; they may even provide no means at all. In these cases players forge their own groups and development teams to create visual tools to analyze game data. Take for instance the exploration games *Minecraft* (Majong, 2011) and *Terraria*. Their gameplay stems from the ability to procedurally generating large, in-game maps that players are meant to explore and manipulate using various digging and building abilities. No map is provided in either game, pushing the

Find Stories in All Genres English

Sort by: VIEWS TODAY 27 Items found

1 2 3 >



Kiss, but don't tell. (Chapter 14)
 Seventeen year old Danielle Pocket is a beauty, and has lots of guys wanting her. Except her protective best friend Brad, and his brother Adam, made them disappear. So when Adam moved away, life for Danielle was good, but now he's back....will trouble stir up? Danielle doesn't want any trouble and she certainly doesn't want to feel the way she does...

By: [Name]
 Created: 05/21/12
 Recommendations: 31
 Views: 268

Fig. 18.6 Players using the Sims Exchange allows players to create their own stories using screenshots from the game *The Sims 3* (© Electronic Arts, Inc.; courtesy of Electronic Arts, Inc.)

player to either remember where they have traversed or to create their own maps. Players have thus created programs to parse *Minecraft* (Minecraft X-Ray 2010 (see Fig. 18.7), *Tectonicus* 2011) and *Terraria* (Terraria Map Viewer 2011) save game files to create visualizations of the procedurally generated maps. Using these maps, players can find the locations of specific objects or other areas of interest, acting as a survey tool for analyzing the topology of the generated maps.

Other examples of visual analytic systems that focus on player behavior analysis are systems built to deconstruct replay files, like those used in the *Starcraft* game series. Replays are files containing “gameplay recorded at high fidelity in order to reproduce as much of the recorded gameplay as possible (Medler 2009).” *Starcraft 2* (Blizzard Ent. 2010) is a real-time strategy game, where players build manufacturing facilities to produce military units and battle one another across an in-game map. *Starcraft 2* record the actions that players perform such as build units, attacking players or even just moving their field of view. Expert players, or players wishing to get better at the game, often revisit their replays in order to analyze their own strategy (much like sports teams analyze video tapes of their games). Normally replay files are reliant on the game’s engine to reproduce the data contained within the files meaning only a single replay can be played in the game at once. However, SC2gears see Fig. 18.8 (SC2gears 2011), a replay analysis tool, takes those replay files and visualizes the information outside of the game environment. The tool also allows players to analyze multiple replays at once, which is not available in *Starcraft 2*. Players can view a 2D graphic visualization of where they built their facilities on the map, monitor their ‘actions per minute’ count, and review the order they built their units and facilities.

Finally, players also band together to create large databases filled with information about specific games. Websites, like WoWhead (WoWhead 2006), which specifically collects data based on the game *World of Warcraft*, contain massive amounts of

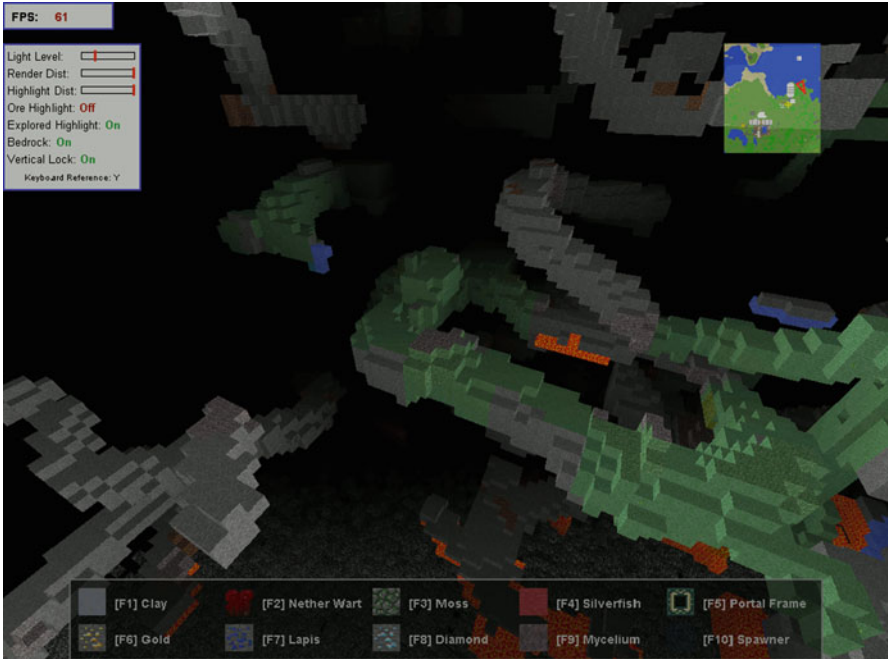


Fig. 18.7 Minecraft X-ray visualizes Minecraft maps to help aid players find specific minerals (Courtesy of Minecraft X-Ray)

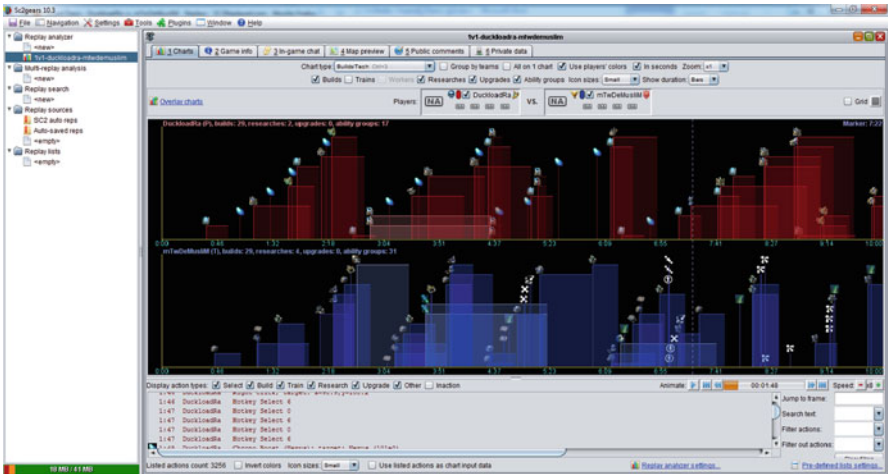


Fig. 18.8 SC2Gears is a player built visual game analytic tool used to analyze replays recorded from Star Craft 2 (© SC2Gears; courtesy of SC2Gears)

information regarding in-game data that relate to concepts like monsters, items, quests, non-player characters, etc. This information is often visualized in relation to maps pulled from the game itself to aid players in finding specific monster, item or quest. *World of Warcraft* itself does not provide such information in the game, which is why players have created this style of game database to function as an optimized way for players to share information about the game and help players that may be stuck or wish to find a specific piece of game content.

18.4 Integrating Visual Game Analytics into Game Development

What has been described so far is how visualization is useful for analyzing game-related data and the types of visual game analytic systems built for particular audiences. For the game developers, third-party services or players who finally decide they need to produce their own visual game analytic tool they must know how to build and deploy their tool. Knowing how the final audience of a tool will react and make use of the tool is just as important when developing a competent visual analytic tool as well. Highlighting these points, this section presents a case study where I helped develop and deploy a visual game analytic tool for a major game studio, Visceral Games. While game developers are not the only audience that visual game analytic tools are built for, as was covered in the last section, they are a vital audience to provide analytic support because they ultimately use analytic tools to alter the development of their game.

The following case study covers the development of a tool called *Data Cracker*, which was built to monitor data collected from the game *Dead Space 2* (DS2). Visceral Games, the developer behind the *Dead Space* franchise, was adding a new multiplayer component to DS2 and wished to use some form of analytic tools to monitor how players were behaving in the game. Data Cracker, therefore, was built to work as a balancing tool monitoring hundreds of thousands of multiplayer matches to assist the DS2 team make balance modifications to their gameplay.

Data Cracker records player events from multiplayer matches, which identify how each in-game player team is behaving. Each match consists of two teams, the human security forces and the monstrous necromorphs, fighting within a confined map. The human team attempts to complete a series of objectives before a timer runs out, while the necromorph team attempts to stop them. The event data that is capture during these multiplayer battles fall in line with data related to the fighting that takes place during each match: win/loss ratios, weapon usage, damage inflicted, traversal movement, kill/death ratios, etc. Both teams have different weapons and abilities that affect how they must approach each battle, too. Thus, one of Data Cracker's goals was to provide the DS2 team with a way to monitor these differences between the two teams and adjust the gameplay when exploits or other undesirable patterns begin to arise from the multiplayer matches. Data Cracker uses the practices and techniques from information visualization and visual analytics to help the DS2 team find those patterns of player behavior. A screenshot of the system is shown in Fig. 18.9.



Fig. 18.9 Built for Dead Space 2, Data Cracker is a visual game analytic tool used to balance Dead Space 2’s multiplayer gameplay (© Electronic Arts, Inc.; courtesy of Electronic Arts, Inc.)

While developing Data Cracker for DS2 our team came upon many insights that we found defined the development of the project, which may be useful for other developers who wish to build their own visual game analytic tools. The insights gained from this project fall into three different categories. First, some insights focused on the production of the tool, which includes when to begin development and the resources that need to be in place after the tool is finished. Second, functional insights cover how the tool was developed. This includes factors like aggregating data for faster load times or partitioning sensitive data records for legal purposes. The third category constitutes insights involving communicating with the game team that demonstrate how important constant correspondence effects the development and deployment of a visual game analytic tool.

18.4.1 Production

18.4.1.1 Game Analytic Tools Should Be Developed While a Game Is Being Developed

Often times when visual analytic tools are built they visualize datasets that have been finalized. Datasets, which are produced by a standard collection procedure, such as airline ticket ordering (Liu et al. 2009), housing prices (Williamson and

Schneiderman 1992), document analysis (Kang et al. 2009), or network structures (Freire et al. 2010). Visual tools used in these cases provide insights about a dataset, which will ultimately not alter the data itself (Kang et al. 2009). Analyzing airline ticket ordering will not alter the data format of those tickets. However, when attempting to build a visual analytic tool for a game developer it is expected that the tool itself may entice the developer to alter not just what data is collected but the format of the data itself. Other times gameplay can be altered simply due to the iterative design process through which games are often developed. For example, the DS2 team routinely swapped out mechanics (such as the number of guns a player can carry or how a weapon damaged an enemy) and features of the multiplayer gameplay while also altering how the telemetric hooks sent back data.

Developers, therefore, can benefit from tools that follow along with this iterative design process. The Data Cracker allowed the DS2 designers to monitor how their game changes were affecting gameplay; additionally, they had to consider the specification for how they wanted to track player's gameplay every time they made a change. Having access to gameplay data forced them to quantify their experiential expectations of each gameplay alteration. During the beta testing phases the DS2 team was able to find weapon exploits using the Data Cracker and subsequently made changes to re-balance the weapons. The team leveraged the tool to aid them during the development phase, and also gain experience with the tool, helping them become even better at analyzing data after the game was released. This is why most game analytic tools should be in place before game teams reach their alpha state. Building the tools before reaching alpha state provides valuable insights into the team's testing phases and give them experience to leverage the tool after the game is released.

18.4.1.2 Early Examples of Visualizations Help a Game Team Understand How Their Game Data Can Be Interpreted and Displayed

Early prototypes are vital for game design, and when building visual game analytic tools prototypes are essential too. For instance, incrementally presenting high fidelity graphs and analytic features helped communicate to the DS2 team how the Data Cracker could eventually be used. Each week the Data Cracker team would introduce new visualizations, describing where the data was coming from and how the visuals helped interpret the data. Instead of using static graphs, the visuals were interactive, similar to how they would function in the tool, and also used real data pulled from DS2. We also used prototypes to win arguments about adding certain features or what data should be collected. For example, after presenting our graphs for analyzing player experience points gained during matches, we were able to show that the points needed to be separated by team type in order to show inequities between the human and necro teams (the two playable "teams" in the game), even though experience points are only awarded after matches where a team had played as both sides. Prototypes brought out these types of situations, thinking of different ways of interpreting the data and what might be missing from our analysis, into our discussions with the DS2 team.

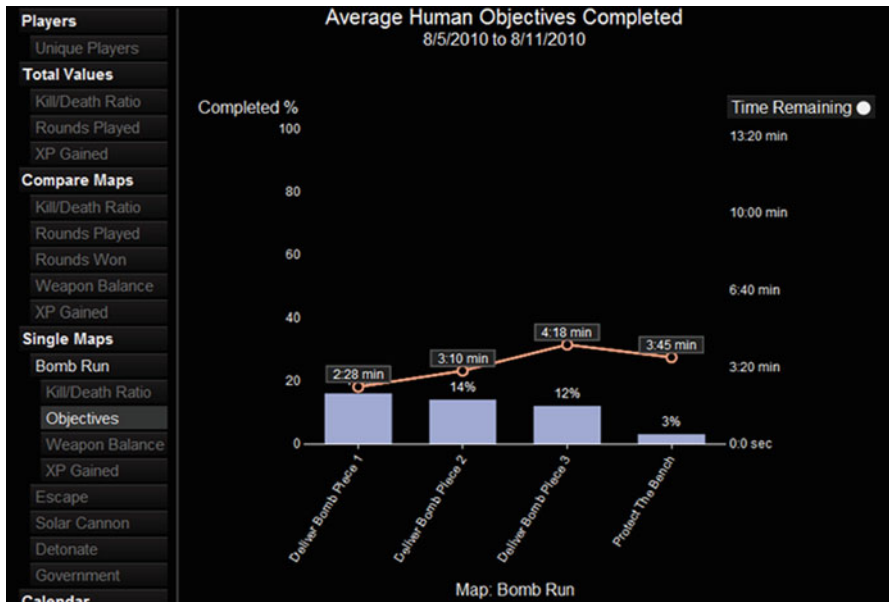


Fig. 18.10 Tiers of graphs are provided in Data Cracker allowing users to analyze data with more or less detail, depending on their needs (© Electronic Arts, Inc.; courtesy of Electronic Arts, Inc.)

18.4.1.3 Provide Tiers of Analysis That Allow Different Audiences Access to at Least a Portion of a Game’s Data

Many disciplines in game development can be affected by the data collected in a game. Capturing player gameplay data may affect designers and producers who alter game mechanics. These alterations can snowball to affect programmers who have to code new mechanics and artists who have to create new content for those mechanics. Therefore, having a tool that is accessible by everyone on the team can allow each discipline the ability to view how the gameplay data is changing and understand why alterations to the game’s design are occurring.

In order to provide a tool for each of these disciplines a method of tiered analysis should be built into a game analytic tool. Each tier provides a different level of abstraction to the game data. With Data Cracker we achieve this by providing graphs that visualize the same set of data but offer different levels of interaction. For example, users can view one graph that shows the total number of rounds played, a second graph that breaks down the total rounds played by map and the win percentage for each team, and a third set of graphs for each map breaking down the percentage of human teams that completed that map’s series of objectives (Fig. 18.10). Game producers may only wish to view the total number of rounds played while a game designer may want to know how often teams are completing a certain objective on a specific map.

Usability tests also proved to be invaluable for creating an accessible tool when our team was iterating on the design of Data Cracker. We conducted tests both

internally with multiple members of the DS2 team and with other EA employees to gauge how quickly they could understand the interface and features. As a result we added features, such as help icons, that explain graphs to users and help indoctrinate team members who had never used our tool before. We also focused on using simple visualizations methods, like bar graphs and line charts, which can decrease data interpretation errors (Heer and Bostock 2010). Certainly, as tool development continues, features that dive deeper into the data can be added but continuing to support the earlier broader features keeps a team focused on the benefits of using analytics.

18.4.1.4 Branding a Tool to a Specific Game or Team Can Create a Greater Sense of Community and May Help a Team Retain More of the Information the Tool Provides

As discussed in the previous section, game developers certainly have many off-the-shelf tools at their disposal. The Data Cracker tool could have been designed to a similar specification allowing multiple game datasets to be analyzed using one common visual interface. However, one study by Bateman et al. (2010) found that by visually embellishing charts and graphs, which to other visualization designers is seen as a distraction (Tufte 1983), people were more likely to retain the information provided in the graphs while simultaneously enjoying the visuals more. Visual embellishment can help strengthen visualizations.

Taking the approach to “embellish”, our tool we set out to design a tool that reflected the personality of DS2. In the end this made sense. Game teams embark on large creative endeavors for months and years at a time; they should have analytic tools that reflect their creativity. While the final version of the Data Cracker did not lavishly embellish graphs, we did mold the appearance to fit into the *Dead Space* art style. The color scheme used throughout the tool was pulled directly from DS2. Blues, for instance, were used to depict data being collected from human players (since blue was often associated with safety in the game) while reds were used for the necromorph’s data (symbolizing danger because necromorphs are the enemies found in *Dead Space 2*). DS2 artwork was also used throughout the tool where it made sense and added style to the different pages. We also made sure to use DS2 artwork when we made promotional material such as weekly data analysis reports. Even the name of the tool, Data Cracker, was a director reference to a concept known as ‘planet cracking’ in the *Dead Space* lore. These added touches created a themed identity for the tool which caused both DS2 team members to get excited about the tool and other game team’s jealous that they did not have their own tool.

18.4.1.5 “Live” Teams Should Be Created to Continue Data Analysis and Tool Development After a Game Is Released

One problem, especially in the game industry, is that after a game ships most of the game team moves on to other projects. This becomes a problem if, after implementing a system for collecting and analyzing game data, no one is around to analyze the data being

collected. A way to change this ‘fire and forget’ behavior is to establish “live” analytic teams to support a game’s analytic efforts after a game is released. A live team is a subset of a game team that is established before the game is released but continues to work on the game after the release, often times creating patches, downloadable content (DLC, additional content associated with a game that can be downloaded separately online) and providing customer support. Not every game team creates a live team, but those who do should include a group of analysts, a group that begins the analytic process before the game ships and is able to analyze the data after the game goes “live.” Analyzing data after a game is released can be beneficial for both the live team, which continues to make content for the players still playing the game, and any portion of the team that may shift their attention to make a sequels or games in the same genre. This sets up a pattern of analyzing data that can then be applied to future games.

18.4.2 Functional

18.4.2.1 Aggregating Player Data into Averages and Totals Make Basic Analysis Faster

The first problem that game analysis run into when collecting game data is often the sheer size of the dataset and querying such large databases of collected data is quite time consuming (Chaudhuri and Dayal 1997). DS2 sold millions of copies and many players took part in the multiplayer matches, each producing their own data stream of gameplay events. The Data Cracker utilized the common practices of Online Analytic Processing (OLAP) (Chaudhuri and Dayal 1997) and Extract-Transform-Load (ETL) processing (Vassiliadis et al. 2002) (see also Chap. 6) to help get around the problem of querying such a large dataset. Calculations that would have normally been very taxing to do multiple times were instead done once a day and stored in a separate database that Data Cracker queries, instead of querying the database containing the raw data. For example, a few graphs had data, such as weapon usage, split up by map. Each morning the Data Cracker ETLs calculated the previous day’s weapon usage statistics for each map and aggregated the data into single elements that could then be read in by Data Cracker. This meant that the ETLs could take many hours to run but the tool itself would never slow down, no matter how many events were collected. The data tables storing the original raw data were also still in tacked allowing for future ETLs to be produce because the raw data was not lost to the aggregation process.

18.4.2.2 Player Gameplay Shifts Over Time and It Is Important to Build Analytic Features That Use Time as a Dimension of Analysis

Time is a key dimension for giving context to gameplay events (Kim 1990); the events we captured for DS2 were no different. The DS2 team wanted the ability to compare different time frames of data which could reflect important periods of time, such as dates of beta tests, the final release date and any dates that corresponded with DLC

releases. For that reason, the timeline came to be the key interactive object in Data Cracker. Every graph in the tool was affected by the timeline and was the one element in the tool that went through several iterations due to usability issues. Our initial design of the timeline consisted of the standard textboxes that allow the user to select a specific date for each and every graph available in the tool. This method proved to be confusing and we altered the tool to allow the user to only set a single time period, which altered all the graphs simultaneously. However, the model of using textboxes to select dates proved to be very slow. Switching to a more interactive timeline bar, which allowed the user to drag and expand their date selected, similar to how timelines work in other tools like video editing software, ultimately became the best solution for selecting date ranges. The large interactive bar also made it possible to place markers which are used to mark key events (e.g. beta testing period) or other interesting dates that may affect the interpretation of that time period's graphs.

18.4.2.3 Analytic Tools Can Help Debug a Game While It Is Under Development

Developing game analytic systems to collect and analyze game data is a software process just like game development, and is therefore prone to errors. Game analytic developers must work with their game team to insure that event hooks accurately collect game data (Kaner and Bond 2004). Once a game analytic tool is established it can function as another form of debugging tool for the game itself. We experienced several occurrences while building the Data Cracker where the data being received was not accurate. For example, at one point we started receiving weapon related data from players who had already finished a multiplayer match, events which should not have been occurring. We had debugged the weapon events some weeks prior and had run our own type of unit tests to determine when weapon events occurred during play. After running further tests, focusing on weapon data, we presented our findings to the DS2 team who found that the game was creating multiple copies of each player during a multiplayer match and each of the player copies were sending gameplay events. Of course, this error was quickly fixed but if that data had persisted it would have seriously skewed the player data being collected. Having methods, such as unit testing, for monitoring what to expect from the event hooks, and determining where they could go wrong, kept the number of errors found to a minimum.

18.4.2.4 Tool Developers Must Be Aware of Legal Issues Affecting Which Countries Player Data Can Be Collected from and Who in the Company Can Have Access to Recorded Data

Collecting player data means game developers must be sensitive to how that data is stored and used. With the recent attacks on many game companies, with the Sony's Playstation Network attack being one of the most publicized incidents (Bilton 2011), it is important that game developers keep player data safe. When developing game analytic systems however, developers need to be able to track a player to

maintain an accurate depiction of the events that took place in the game. Data Cracker attempts to secure this information by separating a player's identifiable data from their gameplay data. A player's username, location and other personally identifiable information are stored on separate servers in relation to any gameplay data that is collected. A unique identifier is then used to link the two datasets, which allows both datasets to be secured and maintained with less of a chance both can be compromised together.

It is also important to note that geographic locations also affect how game companies can collect player data. Compliance with these rules is extremely important in the current climate where online privacy is heatedly being debated (Carvin 2010; Kincaid 2010). There are over a hundred countries where EA cannot collect data. Players in countries, such as the U.S., Canada and Mexico, must agree to Terms of Service documents before game companies can collect their data. In other countries developers can automatically collect player data so long as a player has not opted-out manually. The Data Cracker was able to use EA's online infrastructure to automatically filter out any data that did not comply with the rules laid down by a player's geographic location.

18.4.3 Communication

18.4.3.1 Meeting with Interdisciplinary Teams Ensures the Analytic Features Being Developed Will Benefit the Game Team Later

As mentioned in the second section, game development is an interdisciplinary profession similar to other design and media professions (Kim 2008). Game analytics too requires an interdisciplinary approach especially when game teams often need analytics for different purposes (Medler and Magerko 2011). While developing the Data Cracker, we regularly met with a team consisting of the lead programmer, producer and designer on the DS2 multiplayer team. Each team member had a different expectation for how the tool could aid the team's development process. Balancing the gameplay mechanics was the main goal for the lead producer and designer. The lead programmer aided our team by adding the telemetric hooks into the game and provided his opinions on what data was possible to collect. We ultimately found game bugs and other issues using the Data Cracker, which benefited our main programmer as well. Meeting with an interdisciplinary team meant that the Data Cracker became relevant to multiple members of the DS2 team and helped bypass communication barriers (Haythornthwaite 2005) allowing the team to gain literacy in how analytic tools can contribute to the game development process.

18.4.3.2 Teams May Have Prejudices About Adopting Analytic Tools

Collecting game data is certainly not a new concept and game companies are vocal about their data collection process (Kim et al. 2008; Bungie LLC 2004). One persistent problem, however, is that game teams do not always analyze the data after collection.

Understandably some DS2 team members were skeptical of Data Cracker, because they had seen how ineffective previous game analytic efforts had been. New technology and procedures always bring about this form of cautionary prejudices and speculations (Herbsleb et al. 2002). When developing the Data Cracker, the team's prejudice against game analytics affected the development of the tool, because the DS2 team supplied us with the necessary data from the game. Using the methods previously stated, such as demonstrating prototypes and performing usability tests, we successfully eased the team into using the tool and in turn quashed their prejudice for using game analytic tools. Developers of game analytic tools need to remember that we are currently fighting an uphill battle and only through constant communication, demonstration and usability will game teams accept the tools we are building for game development.

18.4.3.3 An Analytic Team Must Stay Aware of Gameplay Changes

The benefit and challenge of building game analytic tools during a game's development cycle is if the gameplay changes, the analytics change too. Our weekly meetings with the game team not only helped the team become acquainted with the Data Cracker tool but also helped us stay informed of the changes being made to the game. The benefit for having game analytic tools ready early in the development cycle is to allow the team to analyze data being collected from game testers. At the same time the team makes alterations to the game's design, sometimes based on the data being collected and analyzed. Those alterations affect how collected data should be analyzed, and in some cases the tool itself must be altered to match the new changes in the game. During DS2 development various weapons and maps were added or removed from the game. On one level this is cosmetic, changing the name of a gun for example, but in other cases when a map was removed that meant that all visualizations displaying data regarding that map had to be modified too. We, therefore, designed Data Cracker to be modular, visually, with some of the datasets we were capturing from DS2. For example, if a weapon was added the graphs related to visualizing weapon data would automatically populate with the updated list of weapons. Other times the DS2 team would alter the player's abilities in the game, such as only being able to bring in a few weapons into each multiplayer match. This did not change the weapon data format, or any visualizations, but altered how an analyst interpreted the data. Staying informed about the gameplay mechanics and the game's content helped us keep Data Cracker relevant to DS2. While documenting how to interpret the data based on the gameplay changes helped future users of the tool.

18.4.3.4 Analytic Teams Should Act as Part of the Game Team and Continuously Keep the Team Informed

Meeting with the DS2 multiplayer teams helped us keep track of gameplay alterations, but we made the effort to stay in contact with the entire DS2 team. Simple ways of doing this was to talk with other DS2 team members whenever possible

and to participate on internal email lists, generally acting like a fellow team member. Another major way we kept connected to the DS2 team was to distribute a weekly data analysis report detailing the progress of Data Cracker and included interesting information we gleaned from analyzing the DS2 player data. Like the Data Cracker, these weekly reports were DS2 branded, often incorporating artwork from the game, and were designed to look handmade to distinguish them from standard analysis reports that game teams receive from marketing or management departments. Each issue had a certain theme based on an important feature added to the Data Cracker or a specific anomaly we found in the data that week. One weekly issue focused on the Fourth of July holiday weekend (the US national day), for example, where no data was logged from DS2, meaning everyone successfully stayed away from the office. We often received comments that these reports kept people interested in the tool even if they were not affiliated with the DS2 multi-player team or DS2 in general.

18.5 Conclusion

Visual game analytics encompasses more than simply looking at a few analytic tools for visualizing data. It is a domain that spans many theories, fields, tools and systems, which are useful for a variety of audiences. Theories about visual perception and cognitive processing help explain why visualizing data helps us identify patterns while understanding an analytic framework (from gathering data to dissemination) aids in the building of analytic tools. Many groups end up using and creating those tools, including game developers, third-party services and players too. This diversity means analytic tool developers need to understand their audience and the datasets they are delivering. Hopefully, by providing the lessons learned from building the Data Cracker tool, this chapter has shown how an analytic tool developer may approach building a tool for a game team and what data is appropriate to visualize based on a particular audience.

Visual game analytics does not stop here; it is a domain that continues to grow in popularity. During the end of 2011, visual game analytics was in the middle of one of the largest commercial game competitions of the year. Two of the most anticipated games of the year, EA's *Battlefield 3* and Activision's *Call of Duty: Modern Warfare 3* (Cod:MW3) which are both modern military first person shooters, squared off with each attempting to outgun the other. For the most part, the games were compared based on typical gameplay characteristics: single-player and multi-player experiences, visual graphics, and game features like destructible environments. But one comparison had nothing to do with each game's actual gameplay. The developers for both games built systems to record, visualize and disseminate data recorded from multi-player matches: Battlelog for *Battlefield 3* (Electronic Arts Inc 2011) and Elite for *Cod:MW3* (Activision Publishing Inc 2011). Similar to what Bungie.net offers, both are player dossiers systems which track and visualize player data captured from gameplay. Normally

these types of systems go relatively unnoticed by the game community but due to the high profile competition between Battlefield 3 and Cod:MW3 both systems were promoted publicly to gaming audiences. Now questions regarding how these player dossier systems relate to gameplay and how they can be improved have become larger issues. It is easy to predict that due to this exposure many more of the systems will make their way into future games and have a larger role in a player's experience.

As the domain of visual game analytics grows, we will see an even wider array of tools and services. One example is game AI programmers use visualization as part of their debugging process (Dawe et al. 2011). Using visualization to display AI behavior and sensor data, programmers can witness how their AI characters perceive the game world and how those characters are reacting. While Data Cracker was built to visualize player behavior, the behavior of non-player characters can be just as important to interpret and monitor for game developers. As we have seen throughout the chapter visual game analytics has been applied to many game-related datasets and examples like visualizing AI behavior is just an additional way the domain can aid game developers. It is now the task for developers and players to decide how to move this domain forward, to make visual game analytics a prominent feature of future game development and player experiences.

18.6 Next Steps and Resources

This chapter steps through three sections comprised of visualization research, visual game analytic tools already in use and how, in one case, to build a visual game analytic tool. Below are resources associated with each section.

18.6.1 Visualization Researchers and Groups

1. Fernanda Viegas. <http://fernandaviegas.com/>
2. Few 2009. Now You See It: Simple Visualization Techniques for Quantitative Analysis. Analytics Press.
3. Human-Computer Interaction Lab. University of Maryland. <http://www.cs.umd.edu/hcil/research/visualization.shtml>
4. Information Interfaces Group. Georgia Institute of Technology. <http://www.cc.gatech.edu/gvu/ii/>
5. Martin Wattenberg. <http://www.bewitched.com/>
6. Stanford Vis Group. Stanford. <http://vis.stanford.edu/>
7. Tufte 1983. The Visual Display of Quantitative Information. Graphics Press.
8. Ware 2000. Information Visualization: Perception for Design. Morgan Kaufmann.

18.6.2 *Visual Game Analytic Tools*

1. Battlelog. <http://battlelog.battlefield.com/>
2. Bungie.net. <http://www.bungie.net/>
3. Call of Duty Elite. <http://www.callofduty.com/elite>
4. GamerDNA. <http://www.gamerdna.com/>
5. GiantBomb.com. <http://www.giantbomb.com/>
6. Metacritic. <http://www.metacritic.com/games/>
7. Microstrategy. <http://www.microstrategy.com/>
8. SC2Gears. <http://sites.google.com/site/sc2gears/>
9. Social Club. <http://socialclub.rockstargames.com/>
10. Spotfire. <http://spotfire.tibco.com/>
11. Tableau. <http://www.tableausoftware.com/>
12. Terraria Map Viewer. <http://terrariaworldviewer.codeplex.com/>
13. The Sims Exchange. <http://www.thesims3.com/exchange>

18.6.3 *Visualization Toolkits*

1. Protovis (Javascript). <http://vis.stanford.edu/protovis/>
2. jQuery (Javascript). <http://jquery.com/>
3. Processing (Java). <http://processing.org/>
4. Prefuse (Java). <http://prefuse.org/>
5. Flare (Flash). <http://flare.prefuse.org/>
6. Google Visualization API (HTML5). <http://code.google.com/apis/chart/>
7. ScyPy (Python). <http://www.scipy.org/>
8. Weka 3 (Data Mining). <http://www.cs.waikato.ac.nz/ml/weka/>
9. R (Data Mining). <http://www.r-project.org/>

About the Author

Ben Medler, Ph.D., is a Technical Visual Analyst at the Chief Creative Office of Electronic Arts. He received his PhD from Georgia Institute of Technology. His research revolves around using information visualization techniques to build game analytic tools for both game designers and players. He has built visual game analytic tools for Electronic Arts and has developed a proprietary analytic system for capturing data from Flash-based games used for research studies. Ben is pursuing research in a number of related areas using his analytic tools including: understanding differences in player behavior, combining recommendation systems with analytics, and researching the ethics behind gathering player gameplay data. Additionally, he has authored and presented publications on building adaptive game systems, interpreting conflicts in games and linking improvisation with role-playing.

References

- Activision Publishing Inc. (2011). *Elite* (Call of duty: Modern warfare 3). <http://www.callofduty.com/elite>. Accessed 9 Jan 2012.
- Bateman, S., Mandryk, R. L., Gutwin, C., Genest, A., McDine, D., & Brooks, C. (2010). Useful junk? The effects of visual embellishment on comprehension and memorability of charts. In *Proceedings of the CHI 2010*, Atlanta.
- Bilton, N. (2011). Sony playstation network still down after attack. <http://bits.blogs.nytimes.com/2011/04/25/sony-playstation-network-hacked/>. Accessed 5 June 2011.
- Bungie LLC. (2004). Bungie.net. www.bungie.net. Accessed 5 June 2011.
- Card, S., Mackinlay, J., & Shneiderman, B. (1999). *Readings in information visualization: Using visualization to think*. San Francisco: Morgan Kaufmann.
- Carvin, A. (2010). Debate continues around Facebook privacy changes. NPR. <http://www.npr.org/blogs/alltechconsidered/2010/04/29/126394012/npr-listeners-react-to-facebook-privacy-changes>. Accessed 5 June 2011.
- Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1), 65–74.
- Data.gov. (2010). <http://www.data.gov/>. Accessed 5 June 2011.
- Dawe, M., Graham, R., & Schwab, B. (2011). *AI PrOn: Maximum exposure of your debug info*. Presented at GDC 2011. San Francisco: CA.
- Edmunds, A., & Morris, A. (2000). The problem of information overload in business organizations: A review of the literature. *International Journal of Information Management*, 20, 17–28.
- Electronic Arts Inc. (2004). The Sims Exchange. <http://www.thesims3.com/exchange>. Accessed 5 June 2011.
- Electronic Arts Inc. (2010). Autolog. <http://hotpursuit.needforspeed.com/game-info/autolog>. Accessed 5 June 2011.
- Electronic Arts Inc. (2011). Battlelog. <http://battlelog.battlefield.com/> Accessed 9 Jan 2012.
- Few, S. (2009). *Now you see it: Simple visualization techniques for quantitative analysis*. Oakland: Analytics Press.
- Freire, M., Plaisant, C., Shneiderman, B., & Golbeck, J. (2010). ManyNets: An interface for multiple network analysis and visualization. In *Proceedings of the CHI 2010*, New York.
- Ganesh, A. (2010). The essential Chennai rail transport map – Looking back. <http://bitterscotch.wordpress.com/2010/05/07/the-essential-chennai-rail-transport-map-looking-back/>. Accessed 5 June 2011.
- Gerstmann, J., & Davis, R. (2008). Giant Bomb. <http://www.giantbomb.com>. Accessed 5 June 2011.
- Goodman, P. (1993). *Practical implementation of software metrics*. London: McGraw-Hill.
- Hall, P. (2008). Critical visualization. In P. Antonelli (Ed.), *Design and the elastic mind* (pp. 120–131). New York: The Museum of Modern Art.
- Haythornthwaite, C. (2005). Knowledge flow in interdisciplinary teams. In *Proceedings of the system sciences 2005*, Hawaii.
- Heer, J., & Bostock, M. (2010). Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. In *Proceedings of CHI 2010*, Atlanta: GA.
- Herbsleb, J., Atkins, D., Boyer, D., Handel, M., & Finholt, T. (2002). Introducing instant messaging and chat in the workplace. In *Proceedings of CHI 2002* (pp. 171–178), Minneapolis.
- Kaner, C., & Bond, W. (2004). Software engineering metrics: What do they measure and how do we know? *Software metrics symposium, IEEE 2004*, Chicago.
- Kang, Y., Gorg, C., & Stasko, J. (2009). Evaluating visual analytics systems for investigative analysis: Deriving design principles from a case study. In *Proceedings of VAST 2009*, Atlantic City.
- Kantardzic, M. (2003). *Data mining: Concepts, models, methods, and algorithms*. Hoboken: Wiley.
- Keim, D., Andrienko, G., Fekete, J. D., Gorg, C., Kohlhammer, J., & Melancon, G. (2008). *Visual analytics: Definition, process, and challenges. information visualization* (pp. 154–175). Berlin: Springer.
- Kim, S. (1990). Interdisciplinary cooperation. In B. Laurel (Ed.), *The art of human-computer interface design*. Reading: Addison-Wesley.

- Kim, J., Gunn, D., Schuh, E., Phillips, B., Pagulayan, R., & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of CHI 2008* (pp. 443–452). New York: ACM Press.
- Kincaid, J. (2010). This is the second time a Google engineer has been fired for accessing user data. Techcrunch. <http://techcrunch.com/2010/09/14/google-engineer-fired-security/>. Accessed 5 June 2011.
- Liu, Z., Stasko, J., & Sullivan, T. (2009). SellTrend: Inter-attribute visual analysis of temporal transaction data. In *Proceedings of Infovis 2009*. Atlantic City: NJ.
- Manovich, L. (2008). Cultural analytics. <http://lab.softwarestudies.com/2008/09/cultural-analytics.html>. Accessed 5 June 2011.
- Maxis. (2008). Spore API. <http://www.spore.com/comm/developer>. Accessed 12 Jan 2012.
- Medler, B. (2009). Generations of game analytics, achievements and high scores. *Eludamos – Journal for Computer Game Culture*, 3(2), 177–194.
- Medler, B., & Magerko, B. (2011). Analytics of play: Using information visualization and gameplay practices for visualizing video game data. *Parsons Journal for Information Mapping*, 3(1).
- Medler, B., John, M., & Lane, J. (2011). Data cracker: Developing a visual game analytic tool for analyzing online gameplay. In *Proceedings of CHI 2011*, Vancouver.
- Metacritic. (2001). <http://www.metacritic.com/>. Accessed 5 June 2011.
- Microsoft. (2002). X-Box Live.
- Microstrategy. (1989). <http://www.microstrategy.com/>. Accessed 5 June 2011.
- Minecraft X-Ray. (2010). <http://www.apocalypse.com/mincraft/xray/>. Accessed 8 Jan 2012.
- Mochibot. <http://www.mochibot.com/>. Accessed 5 June 2011.
- MoreTerra. (2011). <http://moreterra.codeplex.com/>. Accessed 28 May 2011.
- Nacke, L., Drachen, A., Kuikkaniemi, K., Niesenhaus, J., Korhonen, H., Hoogen, W., Poels, K., IJsselsteijn, W., & Kort, Y. (2009). Playability and player experience research. In *Proceedings of DiGRA 2009*, West London.
- Nonoba. <http://nonoba.com/>. Accessed 5 June 2011.
- Norman, D. (1994). *Things that make us smart: Defending human attributes in the age of the machine*. Reading: Addison-Wesley.
- Orszag, P. (2009). Democratizing data. <http://www.whitehouse.gov/blog/Democratizing-Data>. Accessed 5 June 2011.
- Playtomic. <http://playtomic.com/>
- Pousman, Z., Stasko, J., & Mateas, M. (2007). Casual information visualization: Depictions of data in everyday life. *IEEE Transactions on Visualization and Computer Graphics*, 13(6), 1145–1152.
- Radoff, J. (2006). GamerDNA. www.gamerdna.com. Accessed 5 June 2011.
- Sc2gears. (2011). <http://sites.google.com/site/sc2gears/>. Accessed 5 June 2011.
- Schultze, U., & Vandenbosch, B. (1998). Information overload in a groupware environment: Now you see it, now you don't. *Journal of Organizational Computing and Electronic Commerce*, 8(2), 127–148.
- Shirky, C. (2008). It's not information overload, it's filter failure. Web 2.0 Expo.
- Simpson, C. W., & Prusak, L. (1995). Troubles with information overload—Moving from quantity to quality in information provision. *International Journal of Information Management*, 15, 413–425.
- Sony Computer Ent. (2006). PlayStation network.
- Spotfire Professional. <http://spotfire.tibco.com/>. Accessed 5 June 2011.
- Tableau Desktop. <http://www.tableausoftware.com/>. Accessed 5 June 2011.
- Tectonicus. (2011). <http://triangularpixels.net/games/tectonicus/>. Accessed 28 May 2011.
- Thomas, J., & Cook, K. (2005). *Illuminating the path: The research and development agenda for visual analytics*. Los Alamitos: IEEE Computer Society.
- Tufte, E. (1983). *The visual display of quantitative information*. Cheshire: Graphics Press.
- Valve Corp. (2003). Steam.
- Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002). Conceptual modeling for ETL processes. In *Proceedings of 5th ACM workshop on data warehousing and OLAP*. McLean: VA.

- Ware, C. (2000). *Information visualization: Perception for design*. San Francisco: Morgan Kaufmann.
- Williamson, C., & Schneiderman, B. (1992). The dynamic homeFinder: Evaluating dynamic queries in a real-estate information exploration system. In *Proceedings R&D in information retrieval* (pp. 338–346), New York.
- Wowhead. (2006). <http://www.wowhead.com/>. Accessed 5 June 2011.
- Wurman, R. S. (2001). *Information anxiety 2*. Indianapolis: Que.
- Zoeller, G. (2010). *Development telemetry in video games projects*. Presented at GDC 2010, San Francisco.

Chapter 19

Visual Analytics Tools – A Lens into Player’s Temporal Progression and Behavior

Magy Seif El-Nasr, André Gagné, Dinara Moura, and Bardia Aghabeigi

Take Away Points:

1. Presents a visual analytics system to investigate progression for RTS free to play games.
2. Presents a visual analytics system to investigate progression for RPG games.
3. Discusses and compares the two systems.

19.1 Introduction

As argued in previous chapters, developing engaging interactive new media experiences, including virtual worlds, multi-player or single-player games, involves understanding the target market. Telemetry and metrics can provide a powerful

M. Seif El-Nasr, Ph.D. (✉)
PLAIT Lab, College of Computer and Information Science,
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA
e-mail: magy@neu.edu; m.seifel-nasr@neu.edu

A. Gagné
THQ, Agoura Road, Agoura Hills, CA 91301, USA
e-mail: gagne.andre@gmail.com

D. Moura
School of Interactive Arts and Technology, Simon Fraser University,
Surrey, BC, Canada

B. Aghabeigi, Ph.D.
College of Computer and Information Sciences, Northeastern University,
360 Huntington Avenue, Boston, MA 02115, USA

method to enable designers and other industry professionals to gain insight about their users. Previous chapters have discussed telemetry analysis extensively (see Chaps. 4, 5, 7, 9, 11, 12, 13, 14, 15, 16, and 17). Chapter 18 introduced the Information visualization field and game visual analytics, specifically. In this chapter, we extend the arguments made in previous chapters, specifically investigating the use of *visual analytics systems that reveal player behaviors over time*, where we emphasize the temporal dimension as key to revealing information that indicate causes for specific behavior, and they may also give developers more insight about popular patterns of behaviors.

Previous work within the area of telemetry analysis for games has developed several approaches to this problem, these include:

- using data mining or similar methods to deduce play styles or patterns of play, e.g., the work of Drachen et al. (2009).
- using visualization techniques to visualize aggregate data (such as averages) using charts, e.g., Medler et al.'s work (2011), and
- superimposing spatial aggregate variables over maps, creating heatmaps, e.g., Schuh et al. (2008).

While work within this area is flourishing, it is a relatively new area with several open problems, such as making sense of temporal behavioral data or uncovering behavioral patterns, to mention a few examples that require more research work. In this chapter, we explore the use of techniques to visualize spatial data over time showing designers/analysts more information about player progression. We believe this approach allows designers and analysts to investigate temporal behavioral patterns, which will allow them the deduction of casual effects and complex chains of actions. In addition, like Medler (Chap. 18), we believe that the ability to develop a system that designers and analysts can use to experiment and play with the data for themselves is important due to the fact that designers and analysts bring with them contextual knowledge, which gives them the ability to interpret the results in a meaningful way that can facilitate better design choices. For this purpose, we developed two visual analytics tools: Pathways and Dados, in collaboration with two industry partners: Electronic Arts and Pixel Ante. We tested the former with a designer and the later is still under development.

In this chapter, we will discuss these two systems: *Pathways*, first published in Gagné et al. (2011) and *Dados*, first published in Moura et al. (2011). These systems were developed to specifically look at players' behaviors over time within Real Time Strategy and Role Playing Games, respectively. They were developed to empower designers to make sense of telemetry data through visualization of player data and comparison of behavioral patterns between different players. Our goal is to develop an intuitive and clean visual analytics system that allows analysts to "tell a story" based on the data and generate new hypotheses to be tested. Through developing the two systems in collaboration with two industry partners we were able to, not just deliver two systems, but also investigate and reflect on genre specific mechanics that may be important to capture within a visual analytics system. We will discuss these systems in this chapter; we will also discuss overlaps and differ-

ences between the systems. We believe this to be a good contribution towards a discussion of genre-specific and general-purpose visual analytics systems for games.

19.2 Related Work

Pervious chapters have extensively covered the area of telemetry collection and analysis. In this chapter we will also review some of the visual analytics work, specifically looking at visualization techniques and their use in visualizing game data.

19.2.1 General Visual Analytics and Visualization Systems

Information Visualization and Visual Analytics are fields that have risen from the need to analyze large amounts of data (millions of items). The most relevant visualizations to gameplay telemetry are those dealing with visualizations of events occurring in 2D or 3D space and are continuous in time. The main difference between information visualization systems and visual analytics is that Information Visualization deals primarily with the creation of static visualizations of information. Visual Analytics, on the other hand, provides interactive tools that do not only visualize the information, but also provide selection and filtering of data to support and enhance the analytic thinking process. In this section we discuss several visual analytics tools and systems that were developed for generic purposes to allow analysis of spatial and temporal data. However, we argue that the systems are too generic for use in game environments.

One of the most widely known tools for managing and visualizing geospatial data is *Geographic Information Systems* (GISs). One of the operations one can do in a GIS is to overlay 2D space (i.e. geographic) information over a map; widely known GIS implementations are ArcGIS, MapInfo, and QGIS. Like all major GIS packages, ArcGIS allows not only layering of different types of information on top of maps (say population and smog density) but also the ability to create calculations across or along multiple layers (Drachen and Canossa 2011). Most major GIS packages also have plug-in systems that allow third parties to create industry (Oil, Social Sciences, Environmental research, etc.) specific calculations and visualizations. A GIS of this type can be used to construct heat maps (discussed in earlier chapters) and other visualizations.

For some types of applications, for example games, it is useful to visualize data as a series of events over time (temporal data) (see also Chap. 17). In order to accommodate that, several types of visual analytic systems, including *GeoTime* (Kapler et al. 2007), have been developed. Figure 19.1 shows a screenshot of the system, in this case used to show paths taken by people over space (X-Y axis) and time (Z-axis). *Geotime* is mainly a visual analytics tool focused on analyzing a

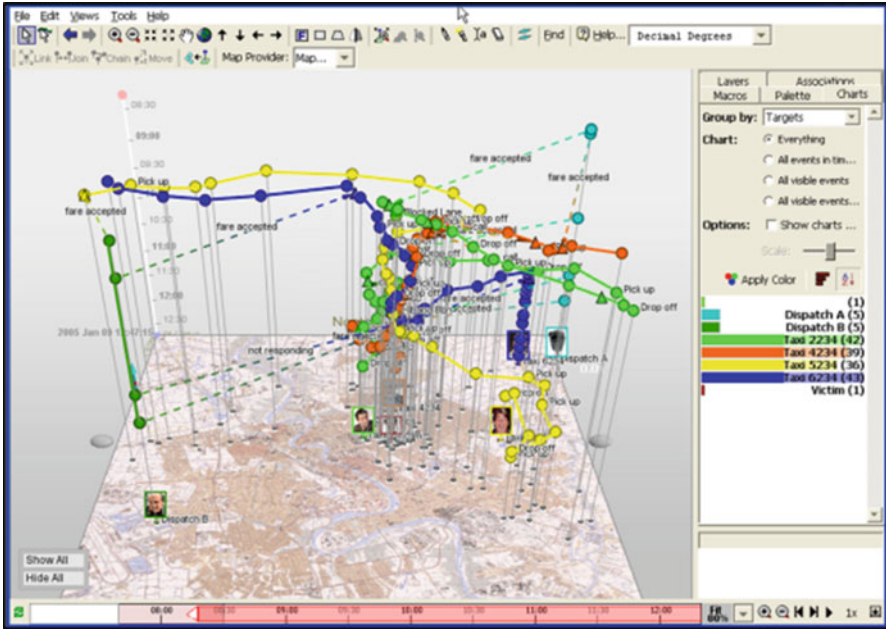


Fig. 19.1 Geotime Screenshot (© 2007 IEEE. Reprinted, with permission, from Eccles et al. 2007)

small set of events to determine possible connections (e.g., between people or when two people met) or cause and effect analysis (e.g., who set fire to a house). As one can imagine with the increase in the number of events over time and due to the focus on understanding how 3D paths overlap, the environment can become cluttered easily, and thus scalability is an issue.

Another well-known visual analytics tool that addressed the issue of interaction with large amounts of temporal data is *TimeSearcher2* (shown in Fig. 19.2) (Kapler and Wright 2005). *TimeSearcher2*'s main components are a timeline selection area in the bottom of the main window, a more detailed plot view in the middle and an even more detailed data view on the right. The interaction between all three levels of visualizations allows for quickly finding events of interest in a large amount of data. This type of visual analytics systems would be useful for gameplay telemetry but *TimeSearcher2* itself is focused on weather data and the use of line plots.

VU-flow is a visualization tool developed by Chittaro et al. (2006) to analyze how users navigate a virtual environment. The purpose was to analyze a large number of users' movement patterns for similarities and outliers specifically trying to find areas of interest. Figure 19.3 shows a screenshot of visualizations created by the system. The movement vector visualization works by breaking a map into a grid and then aggregating all of the movement telemetry data for a specific cell, determining where users within a cell went after they left; the average direction for a particular cell is visualized as an arrow with color representing intensity. This type of visualization could be interesting for videogames as it gives a sense of not just where people were but where they were going. The visualization system discussed in this chapter

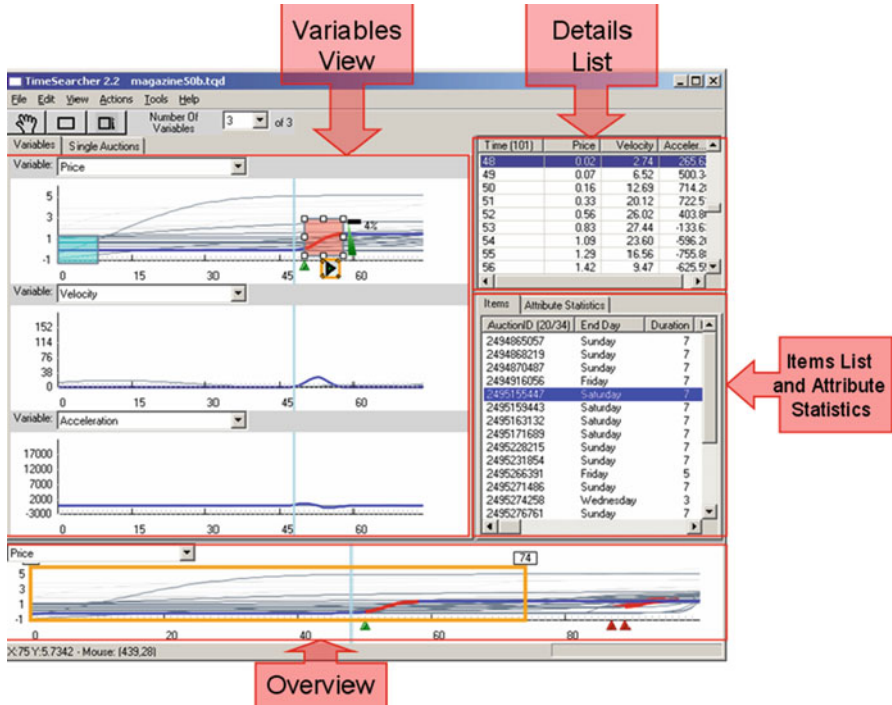


Fig. 19.2 TimeSearcher 3 screenshot (Aris et al. 2005; Hochheiser and Shneiderman 2002; Lusk 2006) (Reprinted with permission from University of Maryland Human-Computer Interaction Lab: <http://www.cs.umd.edu/hcil/timesearcher/>)

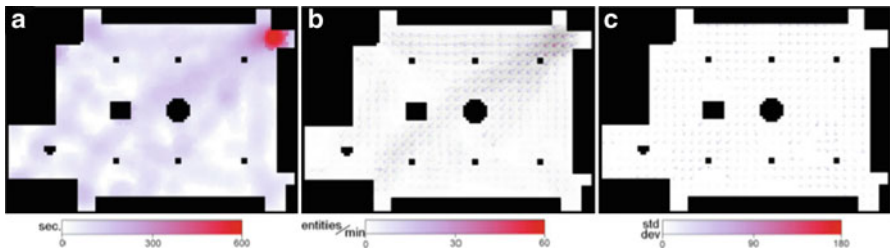


Fig. 19.3 Screenshot from VU-Flow showing “VU-Flow visualizations of data collected for the Udine3D online VE, restricted to the first 40 s of each visit: (a) time spent in the different areas; (b) users’ flow (color indicates flow intensity); (c) users’ flow (color indicates standard deviation of flow direction) (Chittaro et al. 2006)” (© 2006 IEEE. Reprinted, with permission, from Chittaro et al. 2006)

follow similar methods, but embed them within a visual analytics tool and purpose them to game specific uses, as will be discussed later.

Coulton et al. (2008) created a *Geotime*-like visualization for a location based game called *PAC-LAN* (Coulton et al. 2008) to make sense of the players’ strategies. The visualization is similar to *GeoTime* with the Z-axis mapped to time. Unlike *GeoTime*, *PAC-LAN* does not have pre-defined geographic locations in the game.

Correlations in both space and time can be easily seen using this type of visualization, but the ability to select, filter, and drill down on data is not present, since the purpose of the visualization is to explore real-time datasets of a few players rather than datasets with thousands.

Coulton's visualization aside, most of the visualization and visual analytics systems presented here are either too general or focused on datasets that are just far enough away from videogame telemetry to be effectively utilized. For this reason several visualization and visual analytics systems have been developed specifically around the needs of videogame developers. We will discuss these systems next.

In addition to these tools, a general purpose visualization tool that has been used by many industry and academic analysts is Tableau.¹ Tableau enables general purpose visualization techniques utilizing previously researched visualization techniques, including simple graphs and charts to more interactive and spatial visualizations.

19.2.2 *Visual Analytics and Visualization Systems for Games*

The *TRUE* system developed by Microsoft Game Studios (Rashid et al. 2006) was developed to investigate player behaviors, and triangulate in-game telemetry with self report data collected through surveys. User analysts can drill down by clicking on the event icon to view a gameplay video playback. This kind of visualization, while useful in showing if players got lost, does not give the designer an easy way of drilling down and playing with temporal data to understand how users progress through the game with multiple variables involved. In this chapter we present a different approach.

Subsequently researchers have used the *TRUE* system as well as other tools to develop heatmaps. Heatmaps are density maps of specific variables in 2D or 3D space, and thus they are great for showing spatial data. Heatmaps can be used to visually display gameplay aggregate measures, such as player death locations, as shown (see Chap. 17 for more in-depth discussion of heatmaps). More examples of the use of heatmaps include the work on *Halo 3* and *Assassins' Creed* (Drachen and Canossa 2009a; Mahlmann et al. 2010). Although heatmaps give researchers and designers insights, such as unused areas of maps or areas of high death amounts (which may point to level design problems), the aggregation inherent in heatmaps are not appropriate for displaying strategies or for tuning gameplay mechanics involving time or sequence of actions. This is a problem due to the non-linear nature of Real Time Strategy games (RTS) or Role Playing Games (RPGs). Heatmaps are also limited due to the fact that time, not visualized in heatmaps, is of high importance in those domains.

The *Lithium* system (Plourde 2010) was developed as a way of giving real-time visualizations of player statistics. It was developed to support spectators of the competitive multiplayer matches in *Wolfenstein: Enemy Territory*. The primary visualization tool used was a heatmap displaying different statistics (team's occupancy of area, death and healing locations, etc.). In addition, they overlaid data, such as player

¹ <http://www.tableausoftware.com/>

movement (as a trailing line path), current health, and orientation, on top of the heatmaps to give spectators more information. The ability to see multiple events in a single visualization gives observers of a match, not just the overview of the statistic of interest, but also the ability to make inferences about player types (e.g., players that are defensive may only move around in a small location whereas aggressive players move a lot) and their interaction (e.g., you can see one player attacking another). For observers who are interested in the strategy of the players playing the match, the ability to draw the inferences available in the *Lithium* system is very useful.

Skynet (Georg Zoeller 2010), also discussed in Chap. 7, is a videogame telemetry analysis tool that was developed by Georg Zoeller for use with Bioware’s *MassEffect 2* and *Dragon Age: Origins* games. *Skynet* was designed as an all-in-one tool focused on Quality Assurance and Production Management, and as such is focused on tracking the development of the game: the state of stability of various platforms, the number of bugs, and giving in-depth data for bug fixing; any data visualizations developed are aimed at this purpose. Various gameplay variables were tracked for individual sessions, but open-ended exploration of the data visually was not supported. Any analysis of player strategy would have to come from another tool.

The *Data Cracker* gameplay telemetry analysis system was developed by Ben Medler and Jeff Lane for *Dead Space 2* (Electronic Arts, 2010), see Chap. 18. The main purpose was to provide a visual analytics tool for gameplay telemetry that the designers and producers could use to gain an understanding of how balanced the game is. Aggregates of player behaviors were visualized and could be filtered based upon time and preset variables (such as map name). The system is very close to one of the systems we developed (*Pathways*, described next) but is not focused on player’s spatial strategy.

In addition to these systems, similar to the TRUE system, Marsh et al. (2006) developed a system, called *ISIS* (Immersidata analySIS), that collects and synchronizes telemetry data as well as video data to contextualize collected telemetry. They tested it with a serious game. The system is effective for lab studies when one can get video captured data, but does not provide a solution for large amounts of data collected through users playing in their homes.

Using gameplay telemetry for the purpose of exploring player progression through a strategy game or an open-world game has yet to be seriously addressed. None of the visualizations presented so far specifically address this issue. We, thus, present two systems in this chapter that address analysis of player progression where time and choice are important to track and investigate. The main premise is to allow designers and analysts the most interaction with data to allow them to make sense of the data, embedding as much context as possible through visualization.

19.3 Case-Study 1: Visual Analytics System for RTS Games

In this section we discuss an analysis of player behavior in an RTS (Real Time Strategy) game called *Pixel Legions* (Marsh et al. 2006) – a fast paced Flash-based RTS in which players control both a base that produces squads of pixels over time and the squads themselves; the objective for each level is to defeat the opponents by

destroying their base. We first discuss the game then discuss the metrics collected and the visualization tool we developed in collaboration with the game designer. At the end of the section we will discuss the values and lessons we learned through the development of this tool.

19.3.1 *Pixel Legions: The Game*

Pixel Legions encodes the basic mechanics of an RTS. RTS games usually have two main mechanics: the first is the economy, nearly every RTS has some way for the player to gather resources which they can then spend to acquire units; *Pixel Legions* uses the simplest approach: squads of pixels spawn from the base every few seconds (no resource gathering is necessary). The second mechanic is the control of units by issuing orders and locations, which the units in turn fulfill in real time. *Pixel Legions* encodes these basic mechanics, whereby players control a finite resource of squads and can move squads that automatically attack any enemy objects within range. *Pixel Legions* adds several additional mechanics, such as increased damage if two squads are attacking the same opponent from different angles, addition of powerups, hazards, and blocks that push objects in a direction.

The game is comprised of 24 levels. Levels can be skipped and directly jumped to at any time. The first level is a scripted tutorial level introducing the basic mechanics, levels 10, 20–24 are boss levels, the rest reinforce and introduce new mechanics, such as flanking in level 3, powerup locations in level 7 or specific gimmicks, e.g., level 17 forces the bases to continuously move in a circle.

Figure 19.4 shows a screenshot of level 7 of the game, the player (green) starts on the opposite side of the map from the enemy (yellow), the colored squares represent the base where the pixel squads spawn and the white line is a move command given to the base (it originates on the base); the large striped circle in the middle is a barrier that prevent movement through it; and the semi-opaque polygon enclosing two groups of pixels indicates that they are engaged in combat. Figure 19.5 shows a screenshot of level 24.

19.3.2 *Game Analytics for Pixel Legions: Starting with a Set of Research Questions*

Given this game as a case study, we developed a set of questions of interest that were a combination of our own and those from the game designer. These questions fell into two types: *Macroscopic* questions – questions that deal with players actions between matches, and *Microscopic* questions – questions that deal with their actions within a match.

The macroscopic questions were:

- At what point do players stop playing the game?
- How often do players lose a level?
- What aspects of the game do they care about? (completion, submitting/viewing scores, favouring straight-forward levels, gimmicky levels, easy/hard levels)

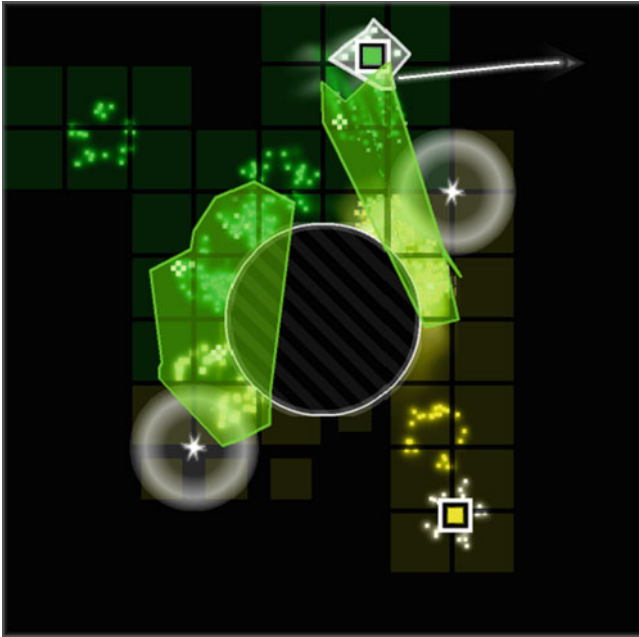


Fig. 19.4 A screenshot of Pixel Legions level 7, the player is in the *upper left* and the enemy the *lower right*. The *white line* represents the path the bases will move along. Moving the base to one of the power pylon circles increases unit production and any units going through the power pylons have upgraded attack power

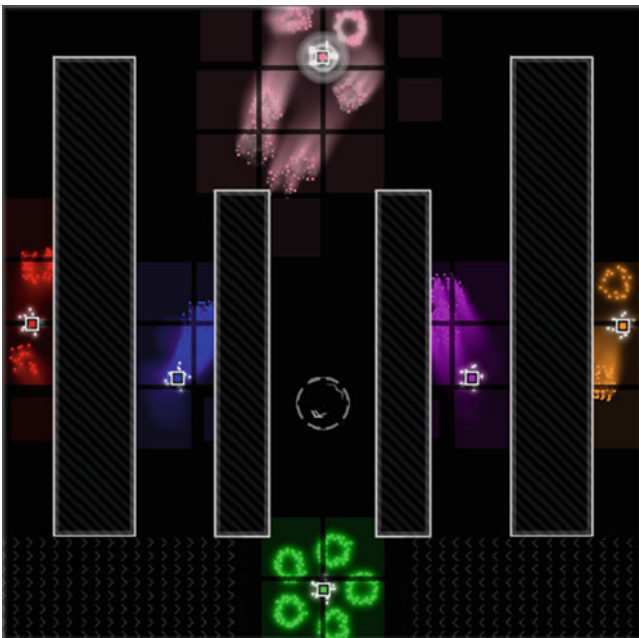


Fig. 19.5 A screenshot of the 24th level of Pixel Legions: a boss battle with the boss at the direct *top (pink)* and the player the direct *bottom (green)*. The *circle in the middle* pushes anything entering it in the direction of the *arrow*

- How do they handle difficulty? (beating levels, skipping levels, bouncing after defeat)
- What levels do people like to play?

The microscopic questions were:

- Are the players doing what the designers expected?
- Are there specific actions that can be associated with wins vs. losses?
- Are players learning how to play a level?

In order to answer these questions, we set up a telemetry system and a number of variables to collect over time.

19.3.3 *Pathways: A Visual Analytics System*

19.3.3.1 Telemetry Collection System

The telemetry collection system was based on the system explained by Niwinski and Randall (2010) and is similar to the one discussed by Drachen and Canossa (2011). It consists of several independent pieces that are interchangeable, a client side telemetry collection API, an apache web server running the PHP message interpretation system, and lastly a database back end to store the data. The entire system is designed to be game genre agnostic to facilitate its use in different game projects.

The client side telemetry collection API was written in Actionscript 3 as the initial target game was Flash based. The real world time on the client side is recorded at the time of the API call. A session ID is automatically generated at the start-up event if one is not supplied to the API. All data is sent to the server via an HTTP page request passing the event name, game name, session ID and arguments supplied to the API call as variables. We used MySQL as the database for this system due to its ease of use and tight integration with apache and PHP. Every game used its own database and every event had its own table.

As the temporal path that a particular player takes within the game was of importance given the questions above, entire sessions were collected rather than a subset of events within a session. Sessions were filtered on a percent-basis with one out of every X recorded (where X is supplied with the client API initialization call and defaults to one).

19.3.3.2 Pathways System Design and Architecture

We designed *Pathways* to aid in the analysis of gameplay telemetry. Notice that the questions discussed in Sect. 19.3.2 are all concerned with time and player progression, learning, and paths before losing interest. Thus, taking these questions in mind, we developed *pathways* with the goal of investigating player behavior over time, where temporal progression is important. Since the game is spatial, then time and space are two dimensions that were important to visualize.

When designers investigate player behavior, they need added functions to control the visualizations, such as manipulate the time range, the sessions being displayed,

selecting spatial areas that are important, etc. These were all important to us as we started to design this system. Therefore, one of the main advantages of Pathways over previous work discussed in Sect. 19.2 is: (a) we added interaction abilities to manipulate the time range of the events displayed in real-time, and (b) we added selection and filtering of sessions and events.

Specifically, Pathways is designed to: (a) visualize videogame telemetry and metric data in a variety of manners, (b) allow designers or users to manipulate different visual elements (such as colour) based on variables within the data, and (c) to be as videogame independent as possible. The domain that is being targeted is 2D or 3D videogames; some assumed properties of these videogames that translate into standard variables for telemetry are:

- The videogames are experienced spatially (they have an x, y, and possibly z position to the player)
- The videogames are experienced over a continuous amount of time (there is a unique ordering of telemetry events that correctly mirror the events in the videogame, often as a *time* variable)
- There is a single play session or individual (the data can be associated to at least a single play session, or at most a single player, often as a *sessionid* or *playerid* variable)

Pathways Architecture

The architecture of Pathways can be seen in Fig. 19.6; currently the system is designed to pull event data directly from a MySQL database into memory in the main system; from there references are saved in a working memory dataset that is

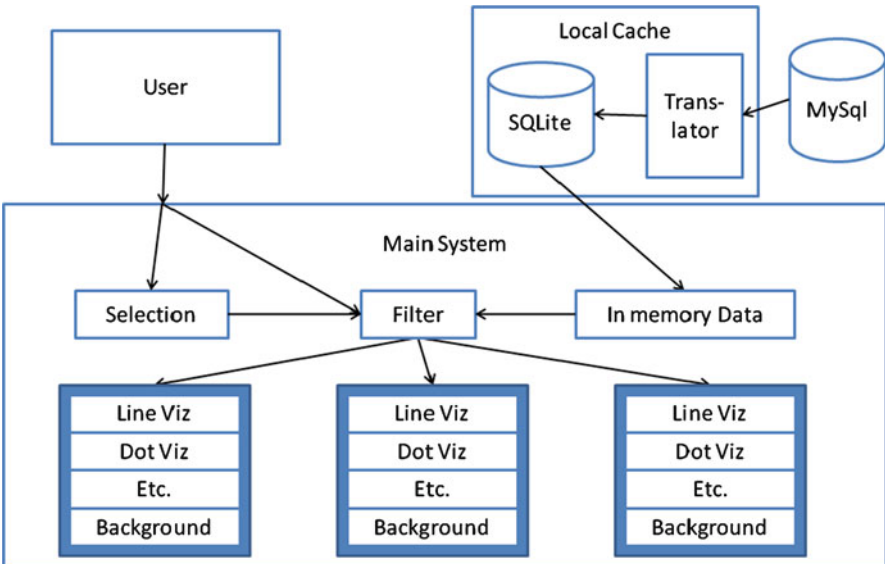


Fig. 19.6 The architecture of Pathways showing the flow of memory and interactions

filtered down to the different panels (blue boxes) each with their own subset (i.e. outcome), each panel then has the same visualization for the same dataset as other panels and a common background. The background is either entirely black or a supplied image that currently must reside in an “images” folder with the *Pathways* executable file.

Having the data stored in the main system means selection and filtering is very easy as filtering based upon the selected item can be done in the main system and then the updated data can be propagated down to the panels and finally the visualizations. A single storage also means that a single record of the current working set needs to be kept and managed, simplifying any code modifying it as well as causing the addition or removal of panels to be much easier.

All mouse events (currently only mouse clicks) are captured by the panel containing the visualizations and then each visualization is queried to determine which, if any, visualized element was selected.

The visualizations currently implemented in the system are:

- *Dot plots*: every event visualized is a circle 2 pixels in diameter centered on the location of the event’s x, y position.
- *Heatmaps*: a map divided into a grid (currently 16×16) where each cell in the grid has a saturation or opacity level depending upon the number of items in the grid. The color of a cell (if a coloring variable is used) changes to whichever color dominates the cell so it will change when more events are added to the cell.
- *Paths*: a line is drawn between two points if, given the total ordering of events in a match, the first point immediately precedes the second. The line has an arrow at the end pointing at the second point to give a visual hint at the direction of the path. If only one point exists for a match no lines will be drawn.

Both the dot and path visualization are visualized at a 20% opacity to exploit the mechanism of overdraw where common actions and behaviors “pop out” from others via the stronger color of the event at the location of multiple actions (the opacity is additive). The variable used for coloring the events can be chosen from the mandatory variables (sessionID, matchID, etc.) as well as non-mandatory variables (teamID, outcome, etc.).

Pathway’s User Interface

An example of the Pathways interface can be seen in Fig. 19.7. The histogram slider (area A) is the slider at the top left of the UI and is meant to give an overview of the events as well as a tool for selecting visualized elements. The histogram will respond to items being filtered out and dynamically shows the number of events per time range represented by one pixel; the smaller the time range, the more minute changes in events over time can be seen. Resizing the window also has the effect of resizing the histogram. All changes in the selected time range are sent to a listening function in the main system that then propagates the changes out to the panels and eventually the visualizations. Note that changes in the selected time range are marked visually

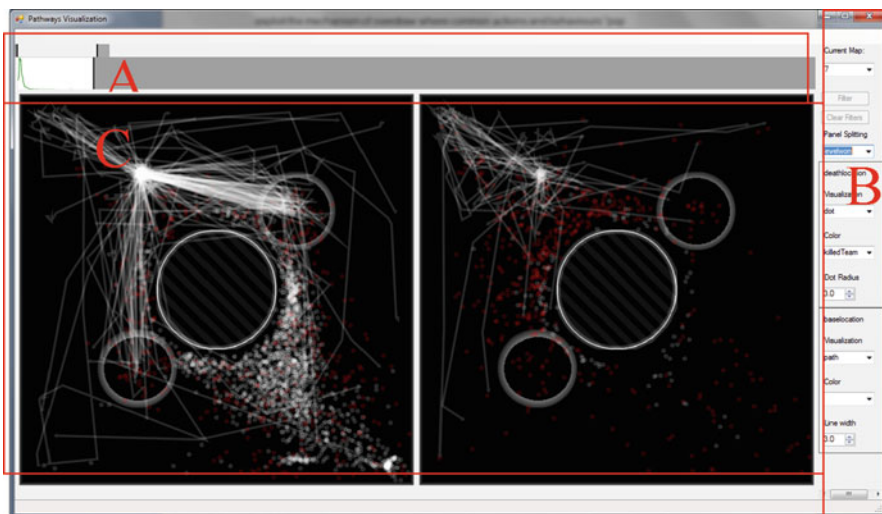


Fig. 19.7 A screenshot of Pathways

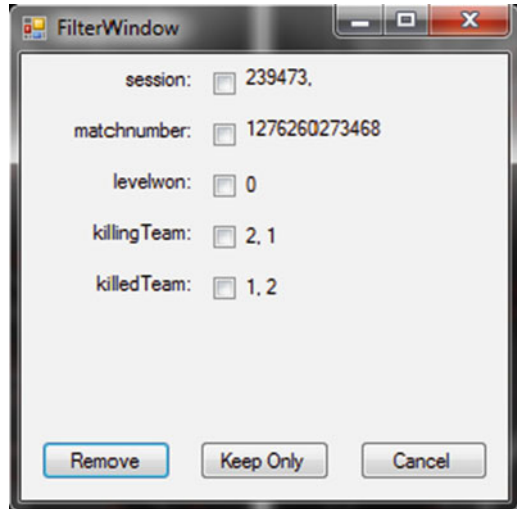
by a white background over the selected area against the normal grey; in addition the thumbs of the slider have a black bar on the side representing the location of time selected.

The options area (area B) contains, in descending order, the map selection box where the user can select any of the available maps; two filter buttons, one for bringing up the filtering window and one to clear all filters; the panel splitting box where the user can select the variable to split the panels upon; and lastly an options panel for each data set displays the visualization type, the variable to color visualized items, and any additional options for the selected visualization type (such as the path width for the path visualization). Both the panel splitting and coloring selections include session id, match id, and time as basic variables as well as a additional variables that exist in the datasets; the panel splitting includes all auxiliary variables (such as variables indicating if the match was won or lost, which team killed the squad, etc.) from every dataset and the coloring selection including the auxiliary variable from the dataset the box belongs to.

The main visualization area (area C) contains the visualization panels. Each panel contains its own layered set of visualizations; the panels are dynamically placed in the visualization area to maximize the amount of drawing space (so 2–4 panels will use 2 panels per column/row, 5–9 will use 3, etc.). If events are found that do not belong to any specific panel they will be placed in their own “none” panel.

The main interactions that are available in pathways are selection, filtering, and timeline change. The filtering window (Drachen and Canossa 2009a, b) allows any of the attributes of the selected events to be filtered upon, either by removing items with selected attributes or filtering out all others (the “filter” and “keep only” options respectively). Filtering works on a temporary working set (not impacted by time

Fig. 19.8 The filter window of Pathways with multiple items selected



range selections) and removing items is cumulative and “keep only” will not contain previously filtered but valid events (Fig. 19.8).

Selection can occur in one of two ways: first is through the user clicking on a visualized element, the single element is selected which causes its information to be displayed in the information bar of the window, it is marked internally as “selected”, highlighted in hot pink (not used in any of the colour palettes) and a reference is kept in the main system. The second way is for a box selection technique (illustrated by Fig. 19.9 through Fig. 19.10): the user clicks and drags the mouse on the visualization (Fig. 19.10), a box is drawn indicating the area of selection and all items within the area are selected, the selection highlighting is updated in real-time to allow the user to know which items are selected as are the references in the main system. Only items in the top visualization (which ever dataset is first in the selection area) are selectable via box selection as it is assumed that in situations with large amounts of items visualized, the user is intending to only select items in the top visualization. In the case of the path visualization, the selected item is the entire path itself, not the individual sub portion (line between two events) that was clicked on.

Filtering can occur after an item has been selected and the filter button has been pressed (note that if nothing is selected the filter button is inactive). When the button is pressed the filter window is displayed (Fig. 19.11) and the session id, match number, or any auxiliary variables in the selected events can be selected; in the example provided, the match numbers are being selected for filtering. Pressing the “Remove” button in the filtering window removes all items that share the corresponding selected values (the effect is cumulative so if match number and session id were selected all events for the selected session ids and any other events that shared the match number would be removed); the visualizations are updated after the filtering has taken place and the histogram timeline is updated. Pressing the “Keep Only” button in the filter window removes all items that do not share any of the selected variable values.

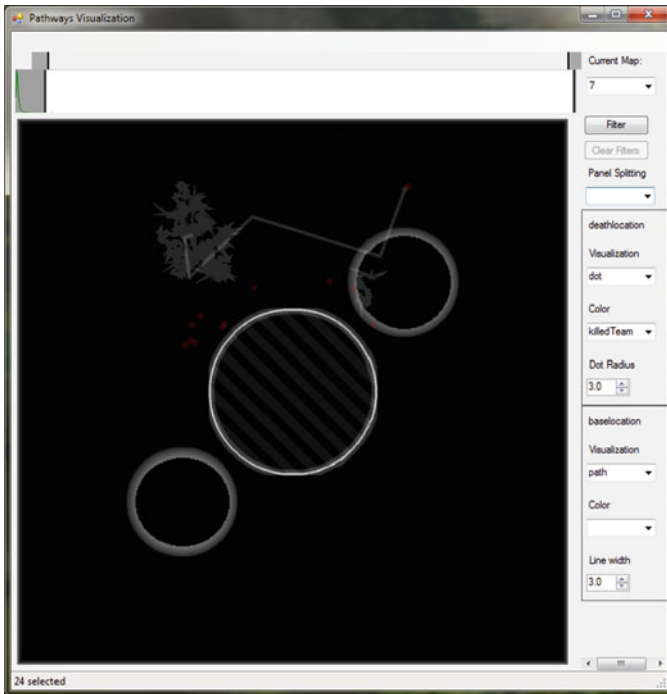


Fig. 19.9 The last several minutes of level 7 visualized in Pathways

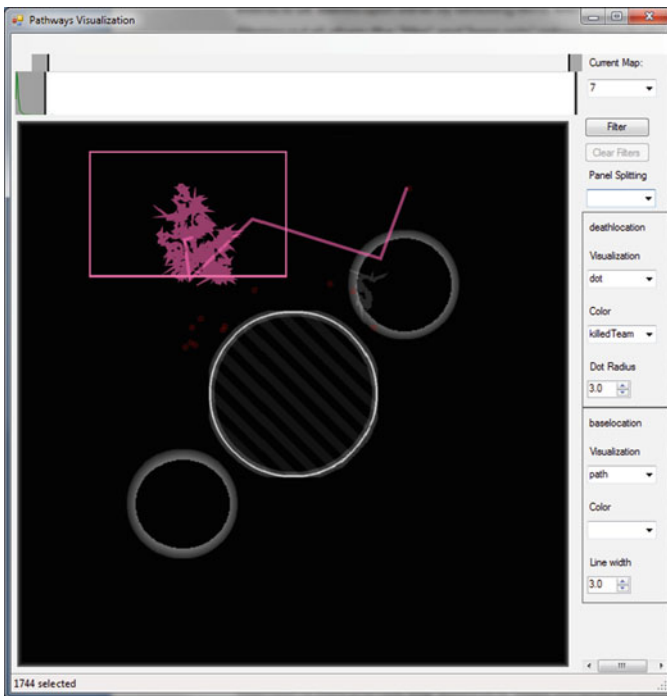


Fig. 19.10 An area selection of base location paths in Pathways

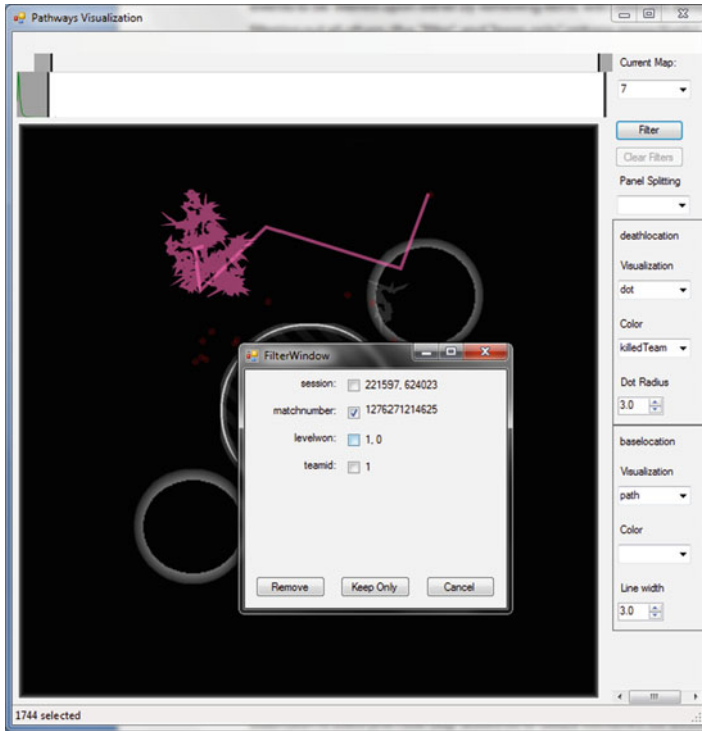


Fig. 19.11 Selecting the match number of the selected items for filtering in Pathways

In order to allow easy application to multiple game types a configuration file system was developed. The basic format of the file is XML but a custom extension is used.

19.3.4 Applying Pathways to Pixel Legions

19.3.4.1 Telemetry Collected

Pathways was used to analyze *Pixel Legions* data. The collection rate was set to 2% of all sessions on the client side to avoid overwhelming the server with data. The system was built with a client side flash API that allows the programmer to send events with a single line of code. The data for the event is then sent to a server, which then inserts it into the database table for that event.

In order to answer the designer’s questions, we collected the following telemetry:

- Level information: level start, level winning, and level skipping
- Movement information: base movement every 5 s to avoid overloading the collection system and slowing the game down

- Death information: squad death including location, time, and team that it belonged to as well as killing team
- Score information: campaign score viewing and submission (*Pixel Legion* does not automatically keep a record of each session’s score)

The dataset analyzed was collected at the time the game was released on the *Pixelante*’s website in June 2011. The remainder of this section focuses on the analysis of the collected data.

19.3.4.2 Results and Analysis

Before we begin the analysis of the questions, we would like to note a limitation. Due to a limitation in the telemetry collection system we were only able to collect data on a per-session basis, not per player; thus we will refer to *sessions* when talking about the collected data and *players* when we refer to possible actions of theoretical players. Given that a single session could play a level multiple times, we will use the term *matches* to indicate the data that may include multiple playthroughs of a single level, and *level* when referring to the specific level of the game.

We applied *Pathways* to the *Pixel Legions* dataset by visualizing base locations over time using a line and arrow element to show the path of the base; squad death locations were visualized using colored dots (colors were associated with the team the squad belonged to); both event types were visualized at 10% opacity to utilize overdraw to give a better sense of what the average behavior was (it would show up darker due to more objects being in the location). Below we discuss the analysis we developed on the questions mentioned above.

Are Players Doing What the Designer Expected?

The major strength of the telemetry is checking if players are playing the game the way the designer intended. To get at this question in *Pixel Legions* we looked at specific levels where mechanics were introduced to see most clearly if players were taking the “correct” actions for the level in order to win. Figure 19.12 is an example of level 7 of *Pixel Legions* visualized in *Pathways*. Level 7 introduces a new gameplay mechanic; specifically it introduces the power pylon as a map object. Moving your squads through pylons makes them more powerful for a short period of time and your base will produce units faster if it is on the object. The locations of the power pylons in level 7 are the white circles in Fig. 19.12.

To arrive at Fig. 19.12 we first started *Pathways* and selected level 7; we then focused on the portion of gametime that had the majority of match data and filtered out the matches that took a very long time; we then split the panel according to the “levelwon” variable and adjusted the time to something near the beginning. Figure 19.13 shows what *Pathways* looks like with the first 44 or so seconds selected; there is more noise in the path visualization in both the winning (on the left) and losing (on the right) matches. We wanted to find the point in time in which there is

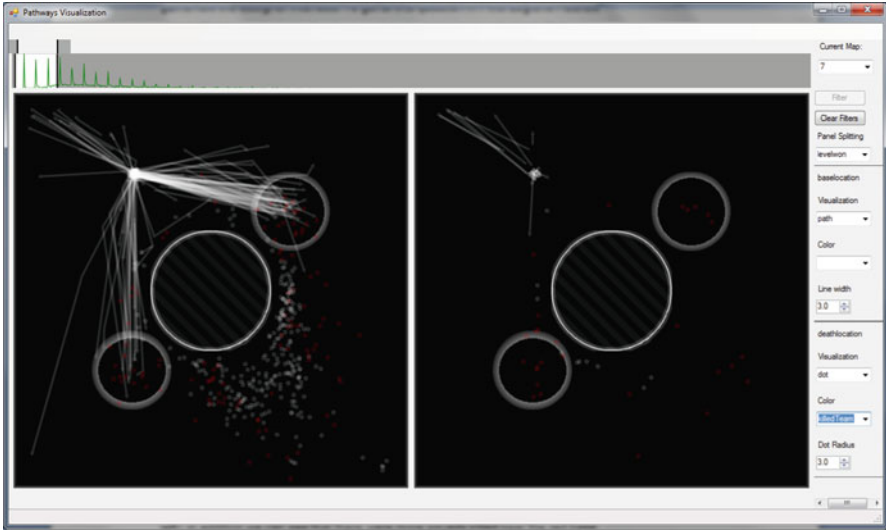


Fig. 19.12 A screenshot of the Pathways visualization system visualizing part of the data from level 7. This represents the first 10 s of gameplay

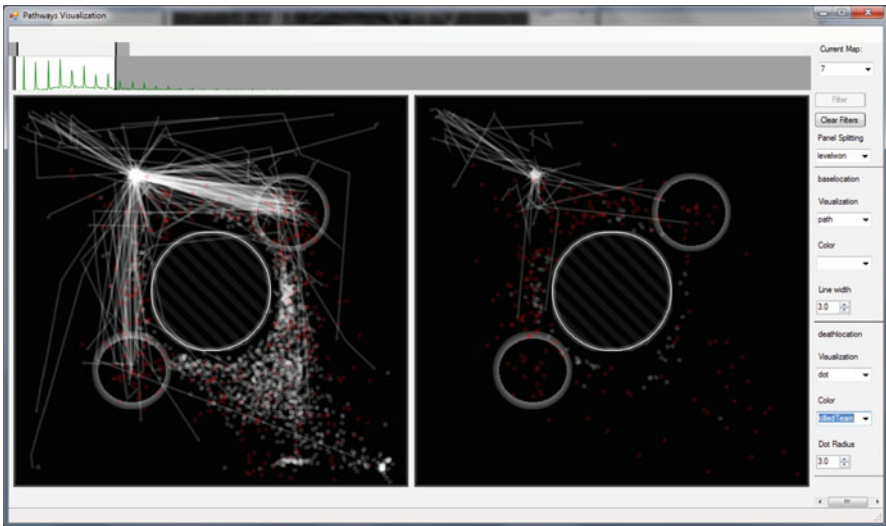


Fig. 19.13 The first 44 s of level 7 visualized in *Pathways*

the clearest distinction between winning and losing matches so we restricted the time range further to the 18 or so seconds as seen in Fig. 19.13.

As Fig. 19.12 is the first 10 s of all matches visualized at once, we can determine that in the matches that were won, players moved to the power pylons earlier (indicated by the mass of white arrows at the pylon’s locations on the left hand side) than the matches that lost (no arrows over the pylon locations on the right); in addition

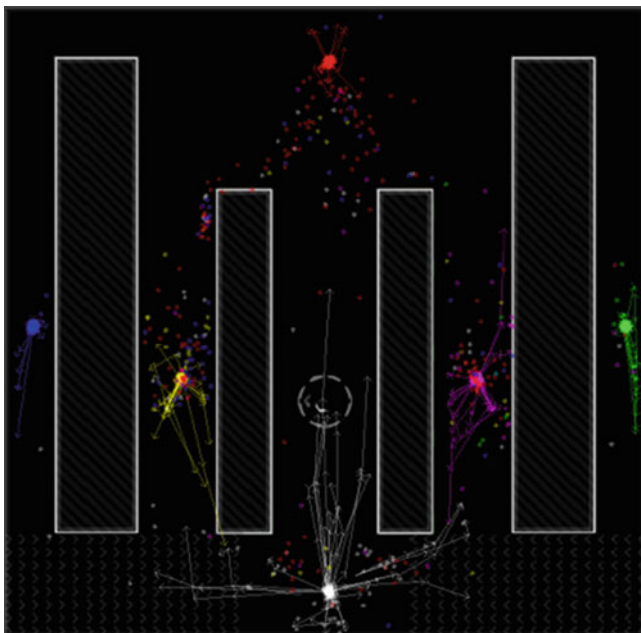


Fig. 19.14 Losing level 24 matches in Pathways

we can see that there were more squads killed near the enemy base’s starting location (opposite side of the map) in matches that won. What this indicates is that players understood that they need to capitalize on these resources fast in order to succeed. The matches that resulted in failure probably were the result of the player being defensive and didn’t capitalize on the power pylons allowing their opponent to overwhelm them. The interesting part about Fig. 19.12 is that even in winning matches players moved their bases into the same defensive position (the lines and arrow pointing to the upper left hand corner of the map); this could be the same player making small, incremental, changes to their default strategy (moving the base defensively) until they won or different players utilizing the same strategy (Figs. 19.14 and 19.15).

Looking at Strategies for Win/Loss

By separating the data according to outcome, winning strategies can be easily seen. Figures 19.4 and 19.5 show the losing and winning views respectively for level 24 visualized in *Pathways* at 25 s into the level. These figures are the two panels from *Pathways*, but expanded to show the paths more clearly. They were constructed using *Pathways* in the same way as investigating whether players were following designer intent and were just as quick to construct. The major difference is in the number of players who tried to move up the middle and sides. Moving up the middle does challenge the red team and possibly move it out of the way but it exposes the flank of the player to the other teams. Moving the player base to the sides places it in a more

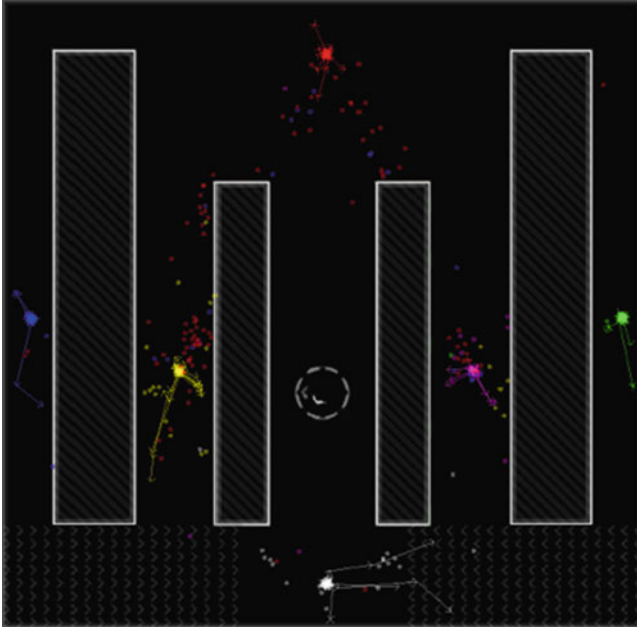


Fig. 19.15 Winning matches of level 24 in Pathways

defensive position by turning the force arrows that would normally funnel enemy squads towards the player base against them, but the same force arrows that protect the players' base also add one more thing the player needs to pay attention to (a number of matches had the base move back to the middle after heading to the side). Keeping the player base in the middle allows the player to minimize the number of exposed sides giving them time to kill off one of the four closer colors, while the player is in the bottom center the boss will also attack the opponents instead of the player.

Learning

The best level to illustrate learning is level 4 as nearly a third (165 of 451) of the sessions had a win/loss ratio of less than one, but also played more than one match (as opposed to level 7's 39 sessions out of 296) and the level is a tutorial level that requires the player to learn the role of a map item (arrows that push things in the direction they're pointing) in order to win. These sessions were chosen because sessions that only played one match wouldn't have a chance to learn and those who won probably understood the trick to the map immediately. Figures 19.16 and 19.17 were created using Pathways in the same manner as described above and took roughly the same time (10–20 s).

Figure 19.16 illustrates the first 10 s of the filtered sessions. Winning matches are on the right, losing on the left. The blue ovals in Fig. 19.16 highlight one key difference between the winning and losing matches: sessions who lost started fighting with the

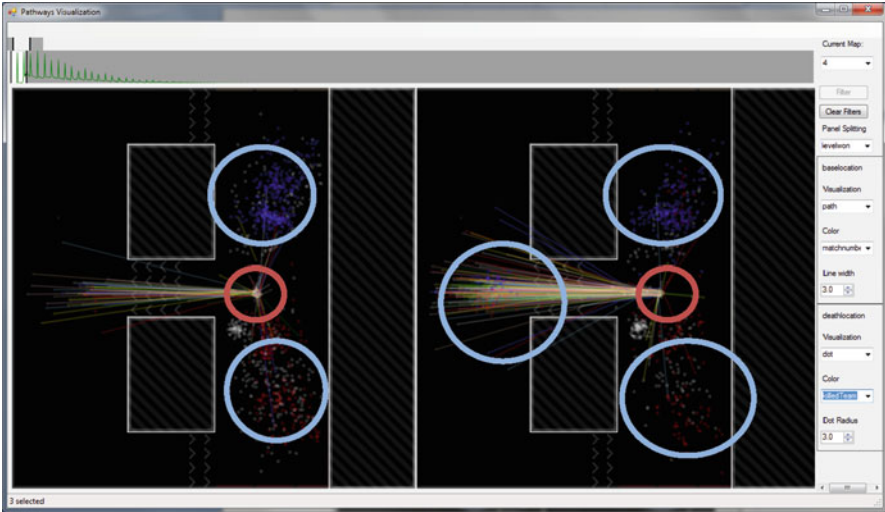


Fig. 19.16 The first 10 s of Level 4 visualized in Pathways, the losing matches are on the *left* while the winning are on the *right*. The larger number of squad deaths in the losing matches suggest that the players behind the matches chose to fight the enemies as opposed to running across the *arrows* directly to the *left*

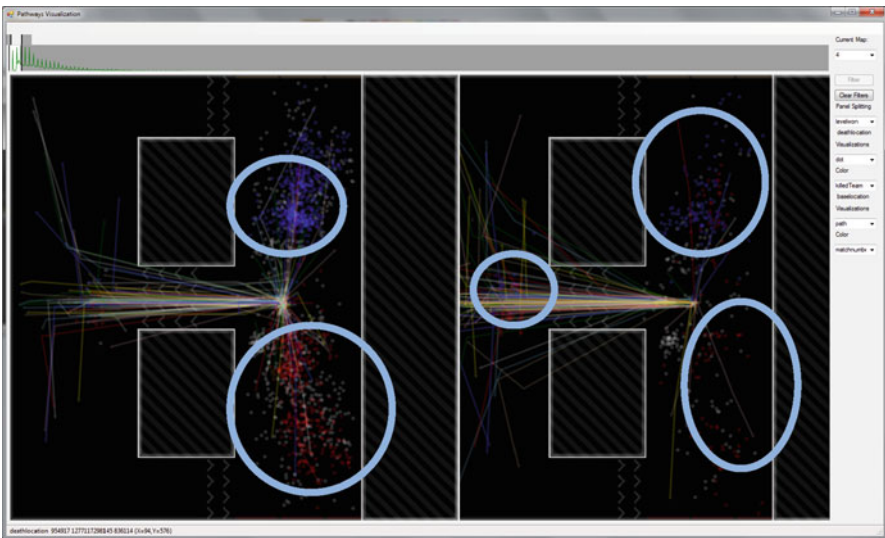


Fig. 19.17 The first 20 s of level 4 visualized in Pathways

enemies much earlier than those who didn’t; in addition, the number of sessions that either stayed at the starting location or moved against the enemies (the brighter splotch of color on the left highlighted by the orange circle) within the same time frame are higher in losing matches.

Figure 19.16 hints that the players who lost were more likely to stay at the starting location and try to kill one of the enemies. This analysis is reinforced by Fig. 19.17 (20 s into the matches), where the number of squads killed near the starting location is much more in matches that lost (the dots within the blue circles) than those that won, hinting that more losing matches tried to fight the enemy AI directly, an interesting point is that in the matches that won we can start to see a growing group of squad deaths just on the left side of the arrows (again, blue circle on the right side).

The caveat of the analysis is that with such a low number of sessions (165) the trends that emerged could just be from a smaller number of players playing the same map over multiple sessions; though the players would have to lose at least once before winning on their second match (and perhaps this means the player didn't learn the level on the first session) to be included in this sample.

19.3.5 Discussion and Limitations

Much of the analysis conducted on *Pixel Legions* is applicable not only to RTS games but any game in which there is a degree of player choice. We would argue that in any multiplayer game there is an interest in what players are doing on a microscopic (within a given match). Revisiting the questions we had:

Microscopic questions:

1. Are the players doing what the designers expected?
Using *Pathways* we were able to visualize all session's plays of a given level, this allowed us to test if the players were utilizing the mechanics introduced and see how quickly they utilized them within a single match. The greatest strength and most common application of telemetry is in answering this question, so *Pathways* would work well for other genres that take place in a 2D space.
2. Are there specific actions that can be associated with wins vs. losses?
We visualized the two outcomes separately in order to identify any specific behaviors associated with each one; we were able to see that in a specific level, moving to a specific location quickly often resulted in a loss. Investigating winning vs. losing behavior is important to many other genres besides RTS as it gets into the question of balance; if a particular item consistently leads to a win then it diminishes strategy and potentially makes the game not fun.
3. Are players learning how to play a level?
We chose a level with a large number of playthroughs that still allowed some level of player choice (level 4). We then filtered out sessions that won in a single playthrough. The visualizations showed that sessions where the winning playthroughs (not the first ones) more closely matched the "perfect strategy" by the designer than the first playthroughs (which lost) indicating that some amount of learning was occurring.

As mentioned earlier, telemetry data doesn't tell us why a session skips; it is a record of a player's behavior; multiple different intents could lead to the same

behavior. It could be that players liked level 12, but without asking the players directly, it’s impossible to know.

As has been mentioned earlier, the fundamental weakness with the recorded telemetry is that ids are for a given session and not a computer or specific player. The possibility of two session ids belonging to the same player means that we cannot determine if the multiple sessions that played through a given level with similar tactics were the same player using their own tactic or two different players who happen to have a similar tactic; in short, the external validity of the sample set is suspect. The lack of player id also made it impossible to tell if the data collected was an actual player or the game designer testing a level.

Although *Pathways* had several limitations, we believe that the system was successful at engaging the designer to get at the questions he wanted to investigate. We have also tested the system with the designer of *Pixel Legions*, who provided feedback enabling us to iterate on the design; thus the system presented was a result of several iterations. Therefore, we believe the system is useful for designers and other researchers and professionals to investigate *temporal* player behavior.

Next we look at another system called *Dados* that we developed in parallel with another industry partner to specifically investigate progression within RPG games. We will discuss the system and outline places where there are overlaps between *Pathways* and *Dados*. We conclude with the lessons we learned through the development of these two systems and some open problems for future work.

19.4 Case Study 2: Visual Analytics System for RPG Games

Unlike RTS where strategy is one of the main game mechanics, RPGs (Role Playing Games) are rich with story, character choices, object and character interactions, fight mechanics, and 3D navigation behaviors. In this section, we discuss a visual analytics system we developed to look into visualization of level and character progression and players’ choices as they progress. This project was done in collaboration with Electronic Arts.

19.4.1 *Dados: A System for Analyzing Temporal Player Behavior Within RPG Games*

For this case study we differentiate between the telemetry collection system and the visual analytics system. The telemetry collection system was developed by Bioware based on a standardized telemetry coding system developed at Electronic Arts. We hooked into their database system to visualize the data. In the following section we will discuss *Dados*, the system we developed to analyze telemetry data from *Dragon Age: Origins*. Since we are still currently working on integrating the system to the huge database of collected telemetry, we will discuss the system and the current challenges we are facing with integrating it to the telemetry system.

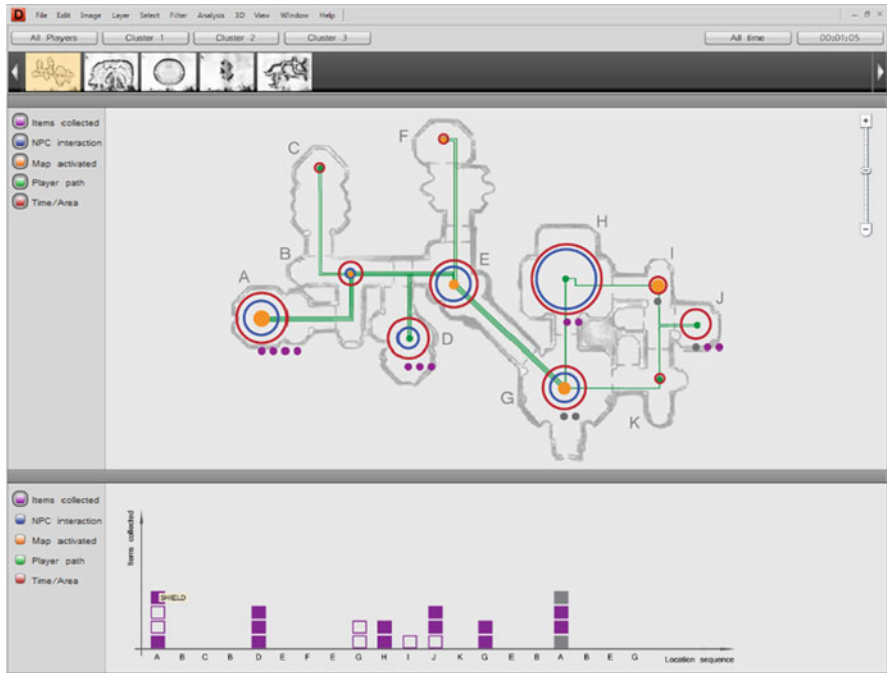


Fig. 19.18 Interface of the proposed visualization system. The first map from *Dragon Age*: Origins is selected from the time line on the top (dark gray bar). All event categories are selected (left menu) and visualized. The bottom graph displays 'Items Collected' in that first map

Like *Pathways*, *Dados* was developed to allow designers/analysts to aggregate, filter, visualize, and compare players' behavior using telemetry data. Figure 19.18 shows the basic interface. Analysts can visualize data from all players or different clusters of players within different time ranges. One specific important difference between *Pathways* and *Dados* is that *Dados* was designed to visualize specific clusters (players with similar play styles), or a single player play through in case of outliers, in a specific time range (see top buttons in Fig. 19.18). Cluster analysis is an important element of any non-linear or sandbox game, as discussed in (Drachen et al. 2012; Thurau and Bauckhage 2010). Bartle (1996) discussed an analysis of MUDs revealing different player types. This article has been recognized within the industry and have led to several designers discussing the importance of understanding player types and including them as part of the design process, thus allowing designers to cater to different motivations of players within their target market. Recognizing the importance of this concept, *Dados* is designed to provide analysts with a method of analyzing player behavior using clusters. This is typically implemented through a clustering algorithm (Bartle 1996). This also provides a scalable solution to the visualization problem (and to *Pathways*), allowing analysts to see more detail on a per player cluster basis.

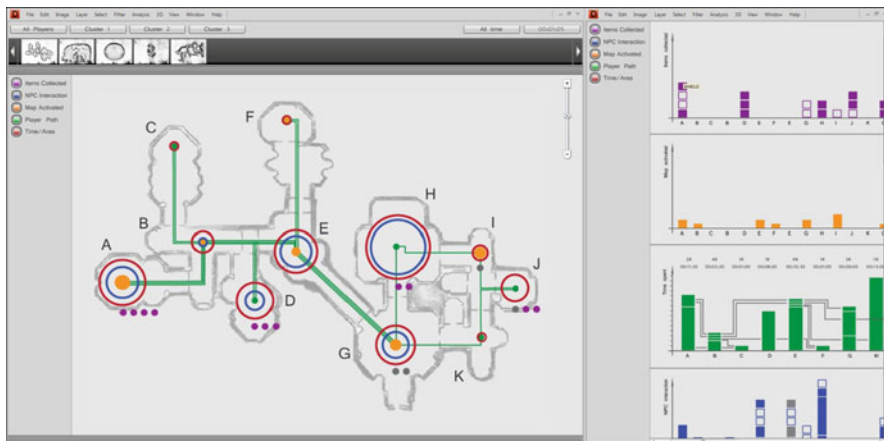


Fig. 19.19 Analysts can minimize the *bottom* part of the interface, expanding the map (*left*). They can open graphs (*or maps*) in new windows (*right*)

In addition to cluster analysis, *Dados*, similar to *Pathways*, shows a time bar that analysts can pan through as they interact with the system. Additionally, *Dados* adds options to show specific event categories, and turn them on and off. In Fig. 19.18, for example, the analyst selected the first hour of gameplay (see top right button). Thus, only the five maps related to that time range are displayed. Analysts can then evaluate if this information is in tune with the designers’ intentions; e.g., did designers expect players to finish five maps in the first hour of gameplay? After the analysts select the map they want to explore, they can choose the events categories they want visualized by using the top menu on the left (Fig. 19.18). Event categories are color coded so it is easy to visually connect them to the information on the map.

Analysts can also open new windows with new data; thus, they can compare player behaviors across different maps. For example, analysts can open different clusters to see what those different groups of players did on the same map. In addition, by right clicking on the bottom left menu, analysts can open graphs that present specific measures (see new window on the right, Fig. 19.19). This is similar to the comparative panels in *Pathways* but specific to RPG information. However, RPG games, unlike RTS, do not have clear win and loss conditions that require different treatments for comparative analysis of behaviors.

Dados adds a panel to give analysts a more abstract view to compare time spent in each type of activity (bottom panel in Fig. 19.18) or the paths a group of players took within a play session. This is important for analyzing paths, which may lead to determining contexts behind quitting, being lost, etc.

Unlike *Pathways*, which recognized specific events that are related to RTS games, such as moving units, attaching, dying, *Dados* encodes several event types that are of importance to RPGs or even adventure games. In the following sections, we will

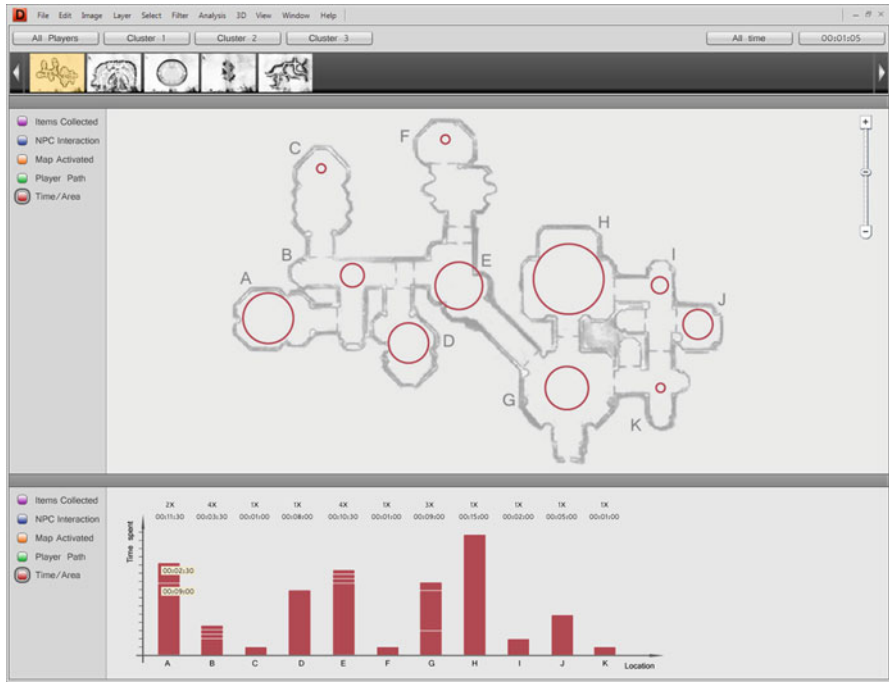


Fig. 19.20 The *bottom* graph shows an example visualization in which players spent 11:30 min within ‘area A’ in total. Players visited ‘area A’ 2 times and spent 9:00 min and 2:30 min in the first and second visits respectively. The amount of time spent in each area is aggregated on *top* of the map

discuss some of these event categories. We chose to focus on five event categories that we think are important for RPG games. However, the system is extensible, i.e. more event types can be added to it.

19.4.1.1 Visualizing Time Spent in Each Area

Knowing how much time players spent playing in a particular area is crucial to hypothesize whether players had problems in an area or not. In our visualization, we detail how much time players spend in each area within a level. First, we named each room with a different letter, where ‘Area A’ is the starting point. A circle represents the total amount of time players spent in one room. The more time players stay in one area, the bigger the circle (Fig. 19.20). The size of the circle is calculated by varying the area, not the radius.

For more precise details about the time players spent in each circle, analysts can open up a graph related to this event category (Fig. 19.20, bottom). In the graph, a different column represents each room. The graph, thus, shows time spent (Y axis) by location (X axis). Analysts/Designers can see how much time players spent in a specific room, and how many times they visited that room. Every time players visit

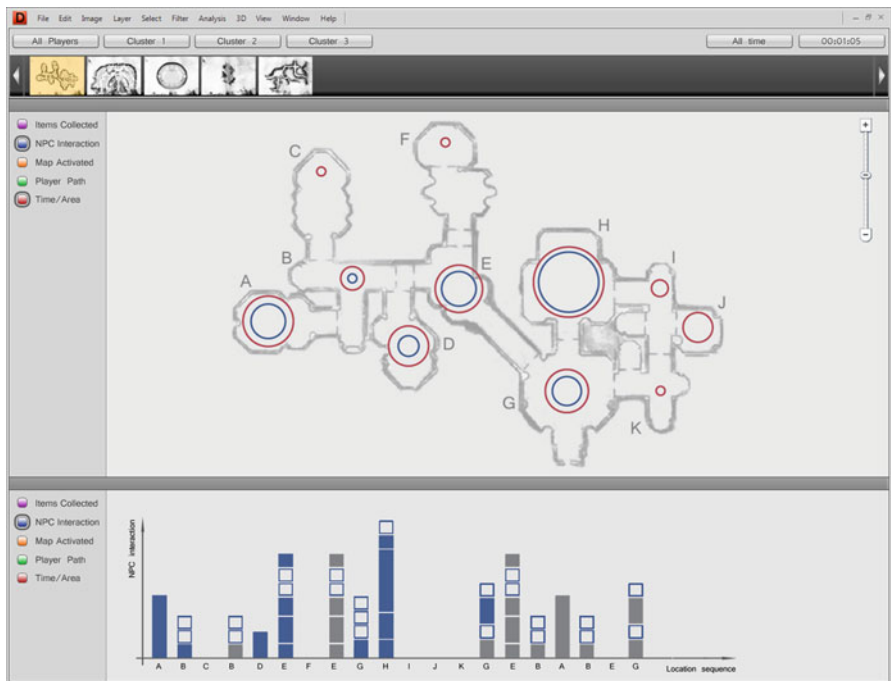


Fig. 19.21 Both, time players spent per area and time players spent talking to NPCs are visualized on the map. On the *bottom*, analysts can visualize how many NPCs are within each room and how many on average were talked to (*blue rectangles*)

a room a new rectangle is drawn in its respective column. The gap between the rectangles shows that players left the room and came back later. For example, in Fig. 19.20, players visited ‘Area E’ four times but spent very little time in the last three times they went there. If analysts mouse over each rectangle they will see the exact amount of time spent on average per each visit.

19.4.1.2 Visualizing Interaction with Characters

Knowing what players do in each area of the game is also important for understanding player behavior. For example, for an RPG, interacting with non-players characters (NPCs) is important because they are a source of information and a key aspect of the game’s narrative. Game designers need to know whether players are interacting with NPCs or not, and for how long.

Figure 19.21 shows such relation. The area of blue circles represent the amount of time players spent on average talking to NPCs in each area. The graph on the bottom shows the locations players visited (X axis) by representations of NPCs interactions (Y axis). NPCs players interact with are represented by blue rectangles. NPCs players do not interact with are represented by white rectangles.



Fig. 19.22 Analyzing when and where players interacted with the map

Gray rectangles represent NPCs players interacted with in a previous visit to a specific area. The graph also shows the sequence of areas visited. In the Fig. 19.21, we can see that the player first went from area A to B, then C. Then, from B to D, and so on. Thus, analysts can track when players talked with any specific character (e.g. one character in A and one character in B despite the fact that there are three characters in B). The size of the rectangle is related to the amount of time players spent talking to an NPC. If analysts mouse over a rectangle in the graph, they will see how much time players spent talking to that NPC on average, and if that NPC was a key NPC or one irrelevant to the story.

19.4.1.3 Visualizing Where Maps Are Activated

In games like RPGs and action-adventure, it is important to know where players open the map and how much time they spend with the map active. Based on this information designers can hypothesize (and then test) if players are getting lost in a specific level or not. The yellow dots on top of the map represent locations and relative amount of time that players spent checking the map (Fig. 19.22). Based on this image, analysts can visualize the most problematic areas in the level. Also, by analyzing the graph on the bottom of the screen, analysts can verify when and where players started checking the map and how much time players spent with the map opened.

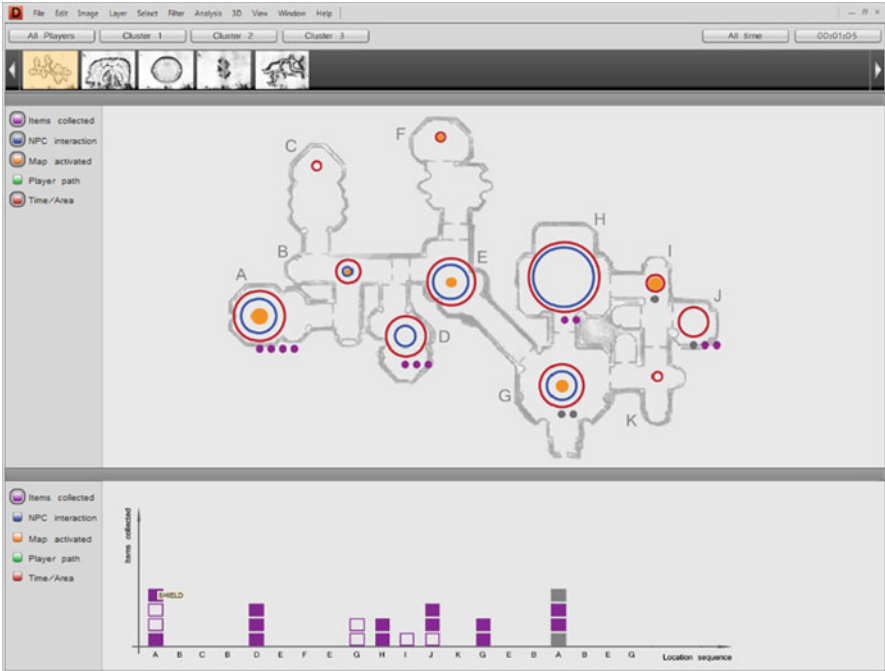


Fig. 19.23 Visualization of items collected in the first map of a game session

19.4.1.4 Visualizing Items Collected

Several games have key items that should not be missed by players. Figure 19.23 shows how *Dados* represents items collected and missed in the game. Each item collected is represented by a purple dot on the map. Dots are placed according to the area the items are located in the level. Gray dots represent items that were not collected by the player. The graph on the bottom of Fig. 19.23 works similarly to the graph of Fig. 19.20.

19.4.1.5 Visualizing Players’ Paths

It is also important to visualize and understand how players navigate through a level as this may shed some light on areas missed or navigation issues. Green lines on top of the map represent paths followed by players (Fig. 19.24). The more players navigate through an area, the thicker the line representing that path will become.

The bottom graph shows locations (X axis) and the time spent in each location (Y axis). Analysts can see how much time players spent in each area and how many times they visited those areas. This bottom graph represents players’ paths in an abstract way, where a column represents each area. Each visit to an area is represented

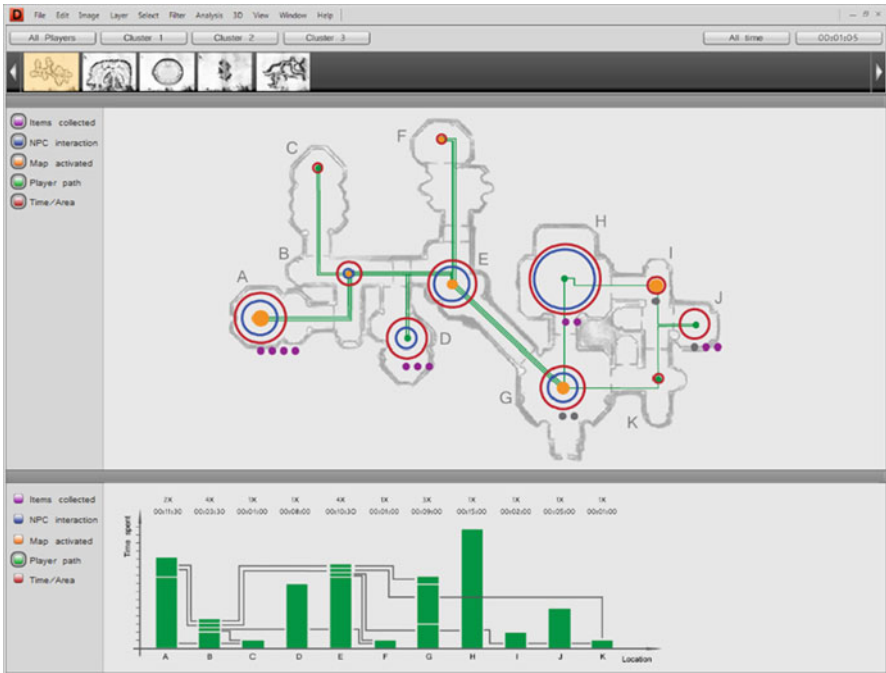


Fig. 19.24 Analysis of player paths and movement through the first map of a play session

by a new rectangle that varies in size based on the amount of time players spent in each visit. Every time a player returns to an area, a new rectangle is placed on top of the first one. Finally, arrows connect the rectangles based on the sequence of locations the player visited.

We note that this is a very different representation from what was discussed in *Pathways*. In *Pathways*, the system was developed to investigate behaviors within the game *Pixel Legions*, in this game maps are not very complex, the whole level is usually visible on the screen. RPG games usually involve more exploration where maps are complex, layered, large and not typically visible in one screen. Some RTS games share this with RPG games, and thus if *Pathways* needs to deal with more complex map structures, some elements from *Dados* may need to be incorporated.

19.4.2 Applying Dados to Dragon Age: Origins

As a first prototype to test the visualization system, we video coded participant data in a lab experiment where we asked five participants to come and play *Dragon Age: Origins* for 30 min. We took the 30-min session and then video coded all the actions that took place in the first map of the game. The video coding procedure was done



Fig. 19.25 The first visualization (*left*) shows a play session that last 2:21 min. The second (*center*), a play session that last 7:34 min. The third one (*right*), aggregates three play sessions (ranging from 5:40 to 6:30 min)

by hand through annotating the video with events that are encoded in the telemetry data, thus developing a list of events with parameters that occur over time. Then we fed the telemetry data into the visualization system. Although this technique is not robust, it was a good first step to see what the progression data can look like and work with designers to establish its value and utility.

In this section we explain the visualizations generated by the five play sessions producing three different clusters where the average times for these clusters were 2:21 (left, $n=1$), 7:34 (center, $n=1$), and 6:00 (right, $n=3$) minutes respectively (Fig. 19.25). Note that these visualizations and completion times are concerned with the first map of the play sessions.

In this first part of the game, areas C and F were locked so players could not get in those rooms. In total, there were 11 items to be collected in this first map and 17 NPCs in the unlocked areas. Players are able to explore the environment, talk to NPCs, open the map (through the menu), and open the menu/inventory. There was no combat in this part of the game.

In the first visualization on Fig. 19.25 (left), the player navigated fast. He did not collect any items and talked to only two NPCs. He opened the map for a short period of time every time he entered a new room until he reached the exit at area G. In the second visualization (center), the player took time to explore the game. He collected two items, visited area D (which was missed by the first player), opened the map, and opened the inventory (light blue circle not mentioned in the previous section of this paper). Finally, in the third visualization (right), the players were apparently more interested in talking to NPCs.

Comparing the first two visualizations we notice that, as players explore the game environment, the visualization becomes busier. In the first visualization, the player left area A as soon as he finished the first dialogue. Note that the red circle (total time in the area) and the dark blue circle (time spent in dialogues) have almost the same area. In contrast, we can see in the second visualization that player 2 spent more time in area A because after finishing the dialogue he opened the inventory (light blue circle) and explored the environment collecting items. Similarly, in the third visualization, players did not leave as soon as the dialogue was over, which gave them time to find one item.

The data showed here comes from only five participants so we cannot draw conclusions at this point. However, using visualizations like these, game designers

can verify whether players are performing in the game as expected. Then, they can test and/or improve the game accordingly. For example, if players are quickly moving from one room to the next, the designer could lock certain areas until the players perform the actions they should (e.g. collect items). If players are taking long time to explore the inventory, designers can investigate whether they are having problems with interface or not. That is, based on telemetry data, designers can construct hypotheses, test and improve their games.

19.4.3 Discussion and Limitations

We are currently in the process of developing a system to hook into the Electronic Arts collected *Dragon age: Origins* telemetry data. However, we are facing several challenges that we developed solutions for but are worth discussing here.

First, as anticipated the data was quite large (over millions of character records) and so data mining is definitely the only option to abstract the data into manageable clusters for visualization. We had over one million users in the Bioware database; these users have played the game on different machine types such as PC, Xbox, PS3. Running any data mining or machine learning algorithms on this huge database will face different problems. For example, clustering can be computationally intensive, which is not appropriate with interactive visualizations where efficiency of producing visualizations is key.

Second, queries in such a database is also problematic as they take a long time to compute, and thus we have to sample. Samples will need to be cross-validated due to issues with sampling, especially with a game with such a divergent game play choices (see the Chap. 9). Specifically, we used a popular sampling technique named Stratified sampling (see Chap. 9 for more information on sampling strategies), to create a smaller data set of 40,000 distinct users, which still represents the whole population very well. To validate findings from this data set, we created another sampled data set with the same size from original data using the same sampling method. This second data set can be used for cross validation of any finding in first data set.

Third, as with any real data before doing any computation on the data, a cleaning step needs to be preformed (see Chap. 12). For example, we noticed that some users had very strange login and log out behavior, they would log out one day, and then come back and play the game after 1 month from the beginning from a different IP address, and in a new game mode. We got suspicious about this fact; we concluded that some User IDs may be shared with different real users, as people can sell or borrow their games. Thus, our assumption that each unique user ID would correspond to a unique player may not be true for all the data we had.

Also, we had users in our database that did not have any corresponding navigation, or game activity records, which led us to the conclusion that some of users' records are missing, which is a common problem (Mahlmann et al. 2010). To solve

this problem, we first removed those users who did not have either any activity records or navigation data which reduced the size of dataset to 23,568 users, then we distinguished players who shared the same User ID (which represents the same license key), by looking at time period delay for each user login and logout behavior and whether they started a new game or not. For example if a user logged out 1 day and came back and logged in after next 20 days, and created a new game, we considered that as a new user, and we assigned him a new user ID. This technique increased the size of the dataset to 36,712 users.

Fourth, user behaviors collected through lab sessions were quite different from the current behaviors we are seeing in the telemetry data. For example, in the lab sessions users are expected to sit and play for the allotted time. However, we found that the frequency of log in and out was significant with the real-world telemetry data. This is important because as users log in and out from a spatial quest based game like *Dragon Age*, their behavior in the first chunk of time when they log back in would be to figure out where they were and how they got there, and thus their behavior will appear as if they were lost. This was interesting since we tried to establish patterns for ‘lost’ behavior and as one can imagine – a lost behavior in a lab session is quite different from that of a frequent log in and out over multiple sessions. We did not visualize log in and out as part of *Dados* above, which was an oversight that we only uncovered when we started working with real data from Bioware. Our next iteration will include log in and out identifiers, which will also need to be taken into account for any data mining algorithm used for predictive analysis or churn based analysis.

19.5 Conclusion

We discussed two different visual analytics systems: *Dados* and *Pathways* developed in collaboration with game developers to look into progression behaviors within RPG and RTS games, respectively. There are several overlaps between the two systems, such as the visualization of paths taken, the inclusion of interaction methods to allow analysts to select and filter with time and space (map) as variables. Clusters are important for scalability and also for player types discovery, and thus both systems needed that. *Dados* encodes clusters as part of its interface and visualization. *Pathways* left that for future work. As maps become bigger and more complex, a way of dissecting them and showing paths of the player to and from regions in the map become important for RPG and also for RTS games. This has been implemented in *Dados* but not in *Pathways* since there was no need to implement it in a simple game such as *PixelLegions*. Analysis of interactions within space was also important, as implemented and discussed in *Pathways* but not currently implemented in *Dados*. Our experience with telemetry from Bioware shows that behavior, especially temporal behavior of players within lab sessions is very different from an ecologically valid environment (such as their homes). Thus, care must be taken when building

visualization systems based on lab telemetry data or playtesting sessions. More work is needed with ecologically valid telemetry. This is a really interesting and very challenging research area; we only scratched the surface with these two case studies. More work is needed to look into other genres, and also investigate the development of more generic systems and tools. We believe these two cases can present an important step towards looking into the development of better visual analytics tools.

About the Authors

Magy Seif El-Nasr, Ph.D. is an Associate Professor in the Colleges of Computer and Information Sciences and Arts, Media and Design, and the Director of Game Educational Programs and Research at Northeastern University, and she also directs the Game User Experience and Design Research Lab. Dr. Seif El-Nasr earned her Ph.D. degree from Northwestern University in Computer Science. Magy's research focuses on enhancing game designs by developing tools and methods for evaluating and adapting game experiences. Her work is internationally known and cited in several game industry books, including *Programming Believable Characters for Computer Games (Game Development Series)* and *Real-time Cinematography for Games*. In addition, she has received several best paper awards for her work. *Magy worked collaboratively with Electronic Arts, Bardel Entertainment, and Pixel Ante.*

André Gagné is a game user researcher at THQ. He received his Bachelor's degree from UBC, Computer Science and his master's from Simon Fraser University. His Master's work investigates the analysis of player progression.

Dinara Moura is a graduate student at Simon Fraser University. She is currently researching player progression and navigation in videogames. She received her Bachelor's degree in Visual Communication and worked as designer for several years. She is also a specialist in Information Design and holds a master's degree in Digital Artifacts Design. Dinara Moura has a passion for human-computer interaction and game user research, cognition and human-centered design.

Bardia Aghabeigi is Game User Researcher and Software Engineer. Bardia has over 4 years of industry experience in software engineering. He has a Master of Science degree in area of Game User Research, and he is doing his PhD at Simon Fraser University in area of HCI Centric Design and Evaluation for Social Networks. He has worked as an intern researcher in companies like Blackbird Interactive, Electronic Arts, Bardel Entertainments Inc, and as software developer in System Groups, and ACE3DFX. He specializes in a mix of data mining, AI, machine learning, and visualization areas. Outside of the industry, Bardia is a passionate hiker.

References

- Aris, A., Shneiderman, B., Plaisant, C., Shmueli, G., Jank, W., Costabile, M., & Paternò, F. (2005). *Human-computer interaction – INTERACT 2005*. In M. F. Costabile & F. Paternò (Eds., Vol. 3585, pp. 835–846). Berlin/Heidelberg: Springer. doi:[10.1007/11555261](https://doi.org/10.1007/11555261)
- Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD Research*, 1(1).
- Chittaro, L., Ranon, R., & Ieronutti, L. (n.d.). VU-flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(6), 1475–1485. doi:[10.1109/TVCG.2006.109](https://doi.org/10.1109/TVCG.2006.109)
- Chittaro, L., Ranon, R., & Ieronutti, L. (2006). VU-flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(6), 1475–1485. doi:[10.1109/TVCG.2006.109](https://doi.org/10.1109/TVCG.2006.109).
- Coulton, P., Bamford, W., Cheverst, K., & Rashid, O. (2008). 3D Space-time visualization of player behaviour in pervasive location-based games. *International Journal of Computer Games Technology*, 2008, 1–5. doi:[10.1155/2008/192153](https://doi.org/10.1155/2008/192153).
- Drachen, A., & Canossa, A. (2009a). Towards gameplay analysis via gameplay metrics. In A. Lugmayr, H. Franssila, P. Näränen, O. Sotamaa, & J. Vanhala (Eds.), *Proceedings of the 13th international MindTrek conference everyday life in the ubiquitous era on MindTrek 09* (pp. 202–209). New York: ACM Press. doi:[10.1145/1621841.1621878](https://doi.org/10.1145/1621841.1621878)
- Drachen, A., & Canossa, A. (2009b). Analyzing spatial user behavior in computer games using geographic information systems. In *International MindTrek conference* (pp. 182–189). New York: ACM Press. doi:[10.1145/1621841.1621875](https://doi.org/10.1145/1621841.1621875)
- Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *2009 IEEE symposium on computational intelligence and games* (pp. 1–8). IEEE: Milano. doi:[10.1109/CIG.2009.5286500](https://doi.org/10.1109/CIG.2009.5286500)
- Drachen, A., & Canossa, A. (2011). Evaluating motion: Spatial user behaviour in virtual environment. *International Journal of Arts and Technology*, 4(3), 294–314.
- Drachen, A., Sifa, R., Baukhage, C., & Thurau, C. (2012). Guns, swords and data: Clustering of player behavior in computer games in the wild. *IEEE computational intelligence in games*.
- Eccles, R., Kapler, T., Harper, R., & Wright, W. (2007). Stories in GeoTime. In W. Ribarsky & J. Dill (Eds.), *IEEE symposium on VAST 2007*, (pp. 19–26). Sacramento: IEEE. doi:[10.1145/1391107.1391109](https://doi.org/10.1145/1391107.1391109)
- Gagné, A., Seif El-Nasr, M., & Shaw, C. (2011). A deeper look at the use of telemetry for analysis of player behavior in RTS games. *International Conference on Entertainment Computing (ICEC 2011) Lecture Notes in Computer Science*, 6972, 247–257.
- Hochheiser, H., & Shneiderman, B. (2002). A dynamic query interface for finding patterns in time series data. In *CHI’02 extended abstracts on human factors in computing systems – CHI ’02* (p. 522). New York: ACM Press. doi:[10.1145/506443.506460](https://doi.org/10.1145/506443.506460)
- Kapler, T., & Wright, W. (2005). Geo time information visualization. *Information Visualization*, 4(2), 136–146. doi:[10.1145/1103520.1103526](https://doi.org/10.1145/1103520.1103526).
- Kapler, T., Harper, R., & Wright, W. (2007). Stories in GeoTime. *Visual Analytics Science and Technology*, 2007. VAST 2007.
- Lusk, E. J. (2006). Email: Its decision support systems inroads—An update. *Decision Support Systems*, 42(1), 328–332. doi:[10.1016/j.dss.2005.01.001](https://doi.org/10.1016/j.dss.2005.01.001).
- Mahlmann, T., Drachen, A., Togelius, J., Canossa, A., & Yannakakis, G. N. (2010). Predicting player behavior in Tomb Raider: Underworld. In *Proceedings of the 2010 IEEE conference on computational intelligence in games* (pp. 178–185). Piscataway: IEEE. Retrieved from <http://www.itu.dk/~yannakakis/cig10TRU.pdf>
- Marsh, T., Smith, S., Yang, K., & Shahabi, C. (2006). Continuous and unobtrusive capture of user-player behaviour and experience to assess and inform game design and development. *1st world conference for fun ‘n games*, Preston.

- Medler, B., John, M., & Lane, J. (2011). Data cracker. *Proceedings of the 2011 annual conference on human factors in computing systems – CHI'11* (p. 2365). New York: ACM Press. doi:[10.1145/1978942.1979288](https://doi.org/10.1145/1978942.1979288)
- Moura, D., & Seif El-Nasr, M. (n.d.). Visualizing and understanding players' behavior in video games: Discovering patterns and supporting aggregation and comparison. *GDC submit on games user research*.
- Moura, D., Seif El-Nasr, M., & Shaw, C. D. (2011). Visualizing and understanding players' behavior in video games/: Discovering patterns and supporting aggregation and comparison. *SIGGRAPH*, Los Angeles
- Nate Hoobler, G. H. (2004). Visualizing competitive behaviors in multi-user virtual environments. *Visualization*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.124.8561>
- Niwinski, T., & Randall, D. J. (2010). Using telemetry to improve Zombie killing. *Game developers conference*, Vancouver, Canada
- Plourde, P. (2010). Designing assassin's creed 2. *Game developers conference*, San Francisco, California
- Rashid, O., Bamford, W., Coulton, P., & Edwards, R. (2006). PAC-LAN. *Proceedings of the 2006 ACM SIGCHI international conference on advances in computer entertainment technology – ACE'06* (p. 33). New York: ACM Press. doi:[10.1145/1178823.1178864](https://doi.org/10.1145/1178823.1178864)
- Schuh, E., Gunn, D. V., Phillips, B., Pagulayan, R. J., Kim, J. H., & Wixon, D. (2008). TRUE instrumentation: Tracking real-time user experience in games. In K. Isbister & N. Schaffer (Eds.), *Game usability advancing the player experience* (pp. 237–265). Burlington: Morgan Kaufmann.
- Thompson, C. (2007). Halo 3: How Microsoft labs invented a new science of play. *Wired Magazine*.
- Thurau, C., & Bauckhage, C. (2010). Analyzing the evolution of social groups in world of warcraft. *IEEE computational intelligence in games*, Los Alamitos.
- Zoeller, G. (2010). Development telemetry in video games projects. *Game developers conference*, San Francisco.

Chapter 20

Interview with Nicklas “Nifflas” Nygren

Alessandro Canossa

Nicklas Nygren is an archetypal independent game developer working in Sweden and Denmark, mostly alone or in very small teams, and specializes in side-scrolling, two-dimensional adventure platformers for desktop computers. He never planned to become a game developer or dreamed to work in a big studio, Nicklas just wanted to make cool games, mostly for himself, just for fun. He was really amazed to find out how many people actually played the games that he created just as a hobby and that he could actually make a living out of it.

Q: Do you make any use of metrics data in your games?

Nicklas: In 2011, during the Global Game Jam, I created this game, *Tikkitt*, where an ecosystem evolved based on player input. It was a very limited use of tracking data from player behavior, and the data was not even sent remotely to a server, all calculations were made on the client, but it was enough to get me interested in the potential of such tool. And now I have decided to incorporate quantitative, remote user testing procedures in all my future productions. As an indie developer, I tend to release several beta versions of my games to the community before the games are released. And I am completely free to do that since I am not bound by the confidentiality agreements and marketing departments that tie the hands of larger developers. I would greatly benefit from tracking user behavior from the hundreds of people that play my games before they are released. With such tool I could easily address the most obvious usability issues to which I am blind, because I am so close to all aspects of production and so proficient at negotiating the challenges and environments that I created. I would discover where players get stuck, what areas or NPCs are too hard or too easy, what is the ideal balancing of difficulty and whether

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

players understand the game features as I intended them. If I had a choice I would prefer traditional, qualitative user testing, observing players while they are engaged with my games, but I do not have the resources to implement such procedures. I do not have the time or the space to invite users over for qualitative assessments. Additionally, the game I am working on at the moment is about 10 hours long and I cannot imagine asking anybody to sit with me for that amount of time.

Q: In your opinion, what are the main benefits of telemetry systems for independent, small developers?

Nicklas: Let me give you an example: *Limbo* (Playdead, 2010) is a fantastic, super-tight game, a great 2 hours long experience. It's easy for the developers of such games to run extensive playtesting, because of the self-contained nature of the game; it can be finished in one session of play. As soon as a game has a larger scope and takes longer to complete, then we start having problems. It's unrealistic for single individuals or even smaller companies to set up longer user testing sessions spanning over several days without almost completely halting production. And the information that I personally am looking for could come just as easily from automated feedback solutions.

Automated, remote and quantitative user research, employing game telemetry systems, is for many small independent developers the easiest, most affordable form of user research. And obviously all tracking systems will be removed from the finished games once they are released, I'm particularly aware of privacy issues. Developing complex telemetry systems may sound daunting and beyond the reach of independent developers, but simple tracking of events, with time and location stamps, like how many people died in that room or how much time players spent in that other room, is actually a very affordable task, even for one-man teams.

Q: Do you envision making use of game telemetry just to optimize user experience or also to maximize monetization opportunities?

Nicklas: I am not interested in freemium or free to play models, it's outside my area. I work within a much more liberal paradigm: people try my games and if they like them, they buy them. So I tend to make the best games I can since it's only the high quality of my games that guarantees a profit. Hence, my interest is only in user experience rather than monetization. Ever since I started making games as a hobby, my main purpose has always been to express something, a nice experience and I just hoped that people would like that. It's never been about maximizing profits, money is a necessary evil that I have to obtain in order to keep making the games I want, while sleeping under a roof with a belly moderately full. If I attempted to include micro-transactions and micro-payments in my games it would kill what I am and what my games stand for.

Q: In terms of visualization and reporting, what is your vision for the tools you want to develop?

Nicklas: All my games until now have been fairly linear, single player side-scrollers, with simple interaction possibilities, not open worlds or multiplayer arenas. Therefore I do not need complex spatial visualizations or dynamic heatmaps. A solid

database and spreadsheets would probably fulfill all my needs. If I have time I would like to invest in developing simple plotting of events such as deaths, over a spatial representation of my game levels.

Q: Do you think that tracking player behavior could have implications for design and for the types of games being produced?

Nicklas: I think it could be interesting tracking player behavior and somehow wiring it as input for some form of procedural content generation system, either using gameplay behavior to shape the game space or to affect and radically change the rules of the game. I don't think that many bigger developers can afford to do that since it begins to mess with the unspoken contract that games have been establishing with players forever: basic rules are taught, learnt and then new content is thrown at the player, until it is mastered again; and then new combinations of rules and content are given to the player until total mastery occurs and the game is completed. Changing the basic rules of a game based on player input, might be too daring for triple A studios that tend to stick to safer bets, and this is precisely the space for innovation that indie developers thrive in. Once a new mechanic or procedure is established and tested, then the large studios can flock in and start adopting it.

Q: You mention that you have decided to incorporate telemetry systems into your next productions, what are the first steps that you will take towards that goal?

Nicklas: First of all I plan to start releasing prototypes and beta versions a lot earlier to the community of players, this way I'd have a large number of players testing my game while I still develop it for a fairly long time. And then of course invest time and resources early on to define key variables to track and strategies to measure those variables, since changing variables halfway through development would render impossible the comparison between different versions. In terms of technical implementation, I prefer to develop myself the metrics tracking, analysis and reporting solutions, even more so because it is going to be fairly simple, but I might evaluate middleware if I find something solid and affordable that meets my needs.

As I mentioned earlier, it is my firm intention to remove tracking systems once the game is released for two reasons: first I do not generally keep working on a game after I decided it's finished, I move on to the next project, so additional data gathered is not so interesting any more. But secondly and most importantly, I have a moral obligation not to invade players' behavior with pointless scrutiny unless such monitoring has an obvious benefit for the player experience. People don't particularly enjoy being monitored and I'd refrain from doing so unless players opted in and would have a tangible benefit for it. Even if it's legal to just track anonymized data, I want to include an “opt in” option during the install and in the settings of the games so players can make a conscious decision whether they want to help me improve the game as much as possible, in the end they paid for it and it should be up to them. Not everything that's legal is ok, just because it's legal. If I can, it is in my interest to respect the rights to privacy of players, if they so wish, giving them the option to escape from a state of constant surveillance, even if just into a game.

Part V

Mixed Methods for Game Evaluation

In this part of the book authors from both industry and academia showcase a number of situations where intelligence gathered from game telemetry is coupled with different datasets and methods. The case studies presented in this part present research questions that can only be answered by triangulating different methods, as they are much broader than the issues that can be tackled by any single method individually increasing dramatically the insights.

This part discusses triangulation techniques looking specifically at three different methods/perspectives and thus the take-aways of the part is connected to these perspectives as follows:

- *Qualitative-quantitative triangulations* explores how metrics data can support traditional user research observations
- *Survey metrics* describes how to create benchmarks and metrics using questionnaires
- *Biometrics* presents state of the art methods to investigate gaze behavior, electrodermal activity, heart activity, brain activity and other forms of psychophysiology.

The part will consist of seven chapters:

- Chapter 21: *Contextualizing Data* introduces mixed-methods methodology and discusses its use in game analytics combining qualitative and quantitative data to get at user research questions of motivation or causes of observed player behavior in the telemetry data. The chapter is a contribution from Eric Hazan is a game user researcher at Ubisoft. He worked on several titles including Assassin's Creed, Splinter Cell, Prince of Persia, Rainbow Six and Ghost Recon.
- Chapter 22: *Combining Back-End Telemetry Data with Established User Testing Protocols: A Love Story* presents yet another case study of combining qualitative and quantitative (telemetry) data to provide meaningful analysis of why players did what they did. The contribution is from Jordan Lynn a game user research at Volition. He performed usability and user experience studies for *Red Faction: Armageddon*, *Saints Row: The Third*, *Saints Row: The Third: Enter the Dominatrix*, among other titles.
- Chapter 23: *Game Metrics Through Questionnaires* discusses the abstraction of metrics data from questionnaires. This is a contribution from Ben Weedon,

principal consultant and manager at Playable Games, a games user research consultancy based in London, UK.

- Chapter 24: *Interview with Simon Møller from Kiloo*, is an interview with Simon Møller the founder of Kiloo, a publisher and independent development company pushing a new model for co-productions. It experienced its first breakthrough with the release of *Frisbee Forever* (Kiloo, 2011) for iPhone, the game topped the charts in nearly 60 countries. The chapter explores the founders perspective on game analytics and mobile development.
- Chapter 25: *Visual Attention and Gaze Behavior in Games: An Object-Based Approach* is a chapter that explores the use of eye tracking and abstracting game metrics through eye tracking. The chapter is a contribution from Veronica Sundstedt, faculty at Blekinge Institute of Technology, Matthias Bernhard, PhD student at Vienna University of Technology, Efstathios Stavarakis, faculty at University of Cyprus, Erik Reinhard, faculty at Max Plank Institute of Informatics, and Michael Wimmer, faculty at Vienna University of Technology.
- Chapter 26: *An Introduction to Physiological Player Metrics for Evaluating Games* is a chapter that introduces affective and emotion measurement through physiology and its use as a method for game user research. The contribution is by Lennart Nacke, faculty at University of Ontario Institute of Technology.
- Chapter 27: *Improving Gameplay with Game Metrics and Player Metrics* explores the use of storyboards a triangulation method for triangulating metrics data from physiology and game telemetry. The chapter is a contribution from Graham McAllister, director of Vertical Slice, a game user research company, Pejman Mirza-Babaei, PhD student at the University of Sussex, and Jason Avent, Disney Interactive Studios.

Chapter 21

Contextualizing Data

Eric Hazan

Take Away Points:

- Player experience in videogames is all about qualitative perceptions of fun and difficulty while player behavior in a videogame can generate any amount of quantitative data. The power of combining analysis from these different sources of data has the potential to yield results that are greater than the sum of its parts.
- Certain branches of the social sciences have used mixed-method research for some years and have recognized its potential for complex analyses. By learning from prior experience of mixing and integrating research strategies, we can improve our understanding of these methods and how they can be used to further develop player experience research methods and processes.

21.1 Introduction

Making games is about crafting software that enables a player to experience fantasies of being an American spy or a space marine, to fight crime or be a mob boss. Players can relive a period in history or unravel a great conspiracy. The ‘experience’ of playing a game is delivered through tests of skills and reflexes. Games can thrill or scare players and they often have a strong story component that is woven into the gameplay, and central to the experience.

Today’s most sophisticated videogames allow players to experience vast immersive worlds. They feature objectives, scenarios, mechanics, and character development. Games help players understand what to do through audio feedback, animations and

E. Hazan (✉)
Ubisoft, Montreal, France
e-mail: eric_hazan@sympatico.ca; eric.hazan@ubisoft.com

rewards. Indeed, hundreds of these features are developed to help a player get through a game, – and enjoy the experience.

As discussed by authors in previous chapters, telemetry as a method for collecting data during the game evaluation process is common in industry and academia. Like many game companies, Ubisoft collects data from multiple sources throughout the various stages of a game’s development. From pre-production playtesting to the gold master and beyond, game developers have access to unprecedented amounts of detailed gameplay data. Instrumentation and telemetry are very powerful because absolutely everything that occurs in a game can be tracked – these are computer programs after all. The challenge is to make sense of the data and to turn it into useful and timely information.

While telemetry systems provide detailed descriptive statistics of what the player did while playing, we often need to understand what motivates player behavior and ultimately, what makes the experience compelling. In 2001, John Hopson (2001) described how controlling when players ‘level up’ in a Role-Playing Game (RPG) can be used to motivate them to keep playing. More recently, Zynga has demonstrated how a game’s design can be informed by using metrics from in-game player behavior (Stern 2011).

One commonly used method of playtesting is to invite external participants to play through portions of pre-release versions of the game. The objective is to see if they experience the game as expected. Imagine you are developing a first-person shooter in a far-west setting. Several players participate in a playtest session where they are asked to walk-through some missions. A few hours into the experience, players reach a difficult passage, where they keep dying in the same spot. Each time they get killed by a stick of dynamite that is thrown at them. They retry, and then fail the mission again in the same way. Quantitative data can tell researchers how players fail, how often, and precisely where on the map. This data can provide a wealth of irrefutable measures. Indeed, a simple bar chart can quickly end hours of debate (“Yes, they died a lot there” or “No, it wasn’t more than other places”). But as players persist in getting through this mission and keep dying, are they enjoying the experience? Are they feeling that the game is being unfair (“Oh no, not again.... I never see it coming”), or are they enjoying the challenge (“I know what I’m doing wrong, and I just need to adjust my timing to get it right”)? In other words, if they were playing the finished product at home, would they stop playing here and never come back?

As researchers, we must validate that these failures are actually a problem. While quantitative data can help to point us in the right direction (many player deaths occurring in the same place), it also tends to be ‘cold’ and lack context. When numbers by themselves are not enough, open-ended questions such as ‘what was fun?’ and ‘what did you find most frustrating?’ can help us gauge how the player is feeling about the experience.

Combining qualitative methods for collecting data with quantitative ones is variously known as mixed-methods research (Burke Johnson et al. 2007) or Multimethodology. Mixing methods to address research questions has been applied in other fields, including the social sciences and health (Collins et al. 2007), and,

more recently, to game development (Davis et al. 2005). In this chapter, I will start by reviewing fundamental theory behind mixed-methods research. I will then review several case studies from game user research. Most of these case studies are gathered through our experience at Ubisoft and two other cases from other user researchers within the industry. Finally, I will conclude with a series of take-away lessons, pitfalls and guidelines to keep in mind when collecting and interpreting gameplay and player experience data.

21.2 Triangulation and Mixed-Methods Research: Key to Games User Research

Much of the research on mixed methodology comes from social sciences and education, where the field of study is often the human factor: motivations and attitudes that inform behavior. Back in 1959, researchers Campbell and Fiske (1959) suggested that mixing methods was a way to accurately measure a psychological trait. Since then, various researchers have looked into mixing quantitative and qualitative methods for collecting and analyzing data to help understand the nuances of particular research questions. In *How to Use Qualitative Methods in Evaluation* (Patton 1987), Patton describes different approaches to data collection and analysis of results to enrich research findings to be more responsive to real-world conditions and to meet stakeholder information needs. He states that, “in practice, it is altogether possible, and often desirable, to combine approaches (p. 62).”

In a 1989 study, Jennifer Greene (1989) and a team of researchers reviewed 57 mixed-methods studies to see how qualitative data and quantitative data were used in combination in evaluating various educational programs. For their purposes, a mixed-methods research design is defined as one that includes at least one quantitative method (designed to collect numbers) and one qualitative method (designed to collect words). In their review, they analyzed each research study, and identified some common characteristics as well as five different ‘purposes’ for mixing methods.

Though this framework for mixing methods described by Greene et al. was applied to research on education policy, this same framework is applicable to games user research. Mixed-methods research studies have some specific characteristics (Greene et al. 1989) that can be summarized as follows:

- The *status* of each method means the relative emphasis given to each type of data collected in the study. In the early stages of a game’s development, where newly conceived ideas, characters and story are developed, qualitative evaluations are more likely to be used. In later stages, or in the post-launch phase, quantitative measures tend to have more weight and importance.
- *Dependence* (or independence) of the methods describes how results from one method can be used to inform the design of the other method. For example, the results of an analysis might generate further questioning and then lead to the development of a follow-up qualitative study.

- The *timing* of data collected from different sources can be combined sequentially or simultaneously, affecting the conclusions that can be drawn and the appropriateness of the results and conclusions. What data needs to be collected and what is the appropriate timing; that is, what do we want to know, and when is it too late?

Greene et al. (1989) also propose five ‘purposes’ for using mixed-methods research:

- *Triangulation*: A major rationale for mixing research methods is the expectation that the results from one method can be evaluated against the results from the other method. For the purposes of this analysis, triangulation refers to *methodological* triangulation, which involves using more than one method to gather data. Triangulation is the convergence of methods that allows the comparison of results, testing one set against another, allowing a more finely tuned understanding of the phenomena. In many cases, one method confirms the results of the other by providing additional proof, which makes the interpretation of the results more certain.
- *Expansion*: seeking to extend the breadth and range of inquiry by using different methods for different research questions. Because the expansion purpose looks to answer separate questions, the scope of the inquiry becomes wider. This broadly defined purpose was the most frequently cited purpose for mixing methods in Greene’s review.
- *Development*: development dynamically uses methods during multiple stages of the study. Each method is implemented sequentially and the results of one analysis are used to shape subsequent methods.
- *Complementarity*. This purpose states that quantitative and qualitative results are used to understand different aspects of the complex phenomena fully. Complementarity seeks to elaborate, enhance, illustrate and clarify the results from one method with the results from the other method.
- *Initiation*. The major aim of this purpose is to explore results that were not predicted. This is an iterative approach that looks for new perspectives in the hope to discover why contradictions exist. In other words, analyzing the results from one method stimulates new research questions or challenges results obtained through the other method.

In the studies that were reviewed, quantitative methods were often used to assess outcomes of the programs, while qualitative methods were used to assess implementation. In Greene’s paper, the researchers concluded that very few of the 57 studies integrated the different method types at the level of data analysis. Although all of the research studies collected qualitative and quantitative data, in almost all the cases the results were analyzed *independently*. However, designing any research study in our field, the best, most informative results come from combining data that measure player behavior, that is what they do in the game, with how they feel about it and why. Integration at the level of data analysis means being able to say: these players made these choices in the game this many times, and this is how they felt about it.

Among the studies investigated in Greene’s research, they found that half of the strategies were guided by expansion or complementarity purposes, and one quarter

by triangulation. Furthermore many of the studies were guided by more than one purpose, often with one purpose being dominant. We found this to be true for studies in game user research as discussed in the case studies below; see also Drachen et al. (2011) and case study 3 in Chap. 14 for more examples.

21.3 Case Studies

Within Games User Research, we also collect data through various tools and methods to gain a better understanding of a phenomenon. In the cases below, some form of telemetry was used in combination with qualitative data to better understand issues with player behavior, emotions and perceptions.

21.3.1 *Splinter Cell: Player Drop-Off*

21.3.1.1 The Game

Tom Clancy's Splinter Cell: Double Agent (Ubisoft, 2006) is an action-adventure stealth game where you play as Sam Fisher, an agent for the NSA. The game, the fourth in the Splinter Cell series, takes you through 11 missions in various locations around the world as you infiltrate a gang, gain their trust and then attempt to break them up. A screenshot of the game is shown in Fig. 21.1.



Fig. 21.1 Screenshot from *Tom Clancy's Splinter Cell: Double Agent* (© 2006–2010 Ubisoft Entertainment; courtesy of Ubisoft)

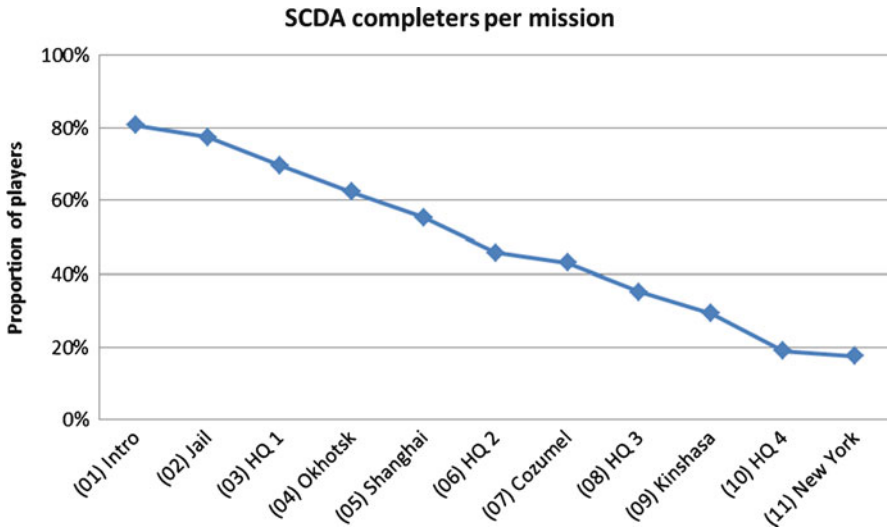


Fig. 21.2 SCDA completers per mission

21.3.1.2 The Data and Analysis

After the game launched in fall 2006, various data were collected to see how far players advanced through the missions and how many completed the game. Some of the data collected included which missions were completed, how many times players retried them and the time spent playing each mission.

Figure 21.2 shows the mission drop-off. Along the x-axis are the missions, in chronological order. The y-axis is the percentage of players who reached each mission but stopped playing there. This graph shows the stage at which players stopped playing; culminating to 18% of those who started that completed the final mission.

Presented with these results, we wanted to explain why the game had this somewhat low level of completion. What caused players to give up as they progressed through the missions? To explain this phenomenon, the first hypothesis was that the missions were too difficult, and after too many tries, the players would abandon the game. To test this, we looked at the number of save-game loads per mission as a measure of difficulty. Each time a player loaded a saved game, it was a result of either starting a play session or retrying after failing. We also had to normalize for the length of missions. The exact measure we used was, for each mission, the number of save-game loads per hour of gameplay. This is represented by the dotted line in Fig. 21.3. The solid line shows the drop-off rate that is the proportion of players who stop playing in each of the missions.

The first mission typically shows a high drop off (solid line). Thus, we concentrated our analysis on the missions with the highest drop-off rates. Most missions

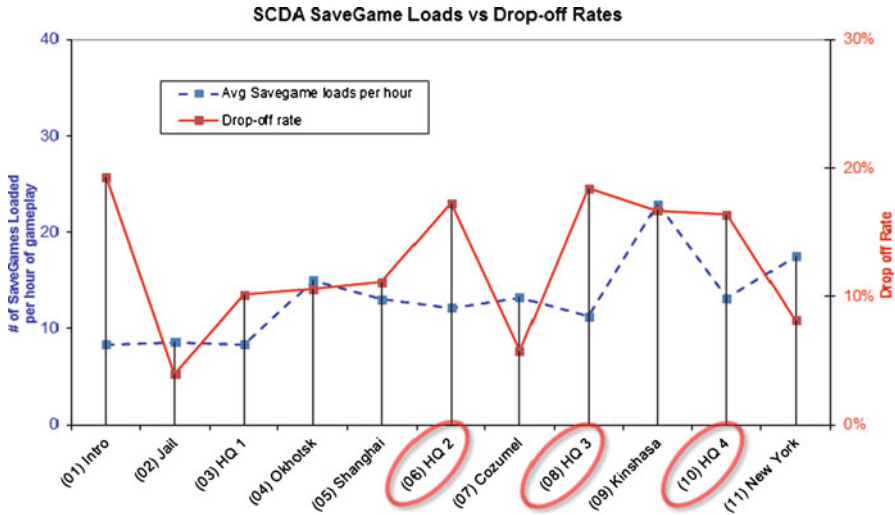


Fig. 21.3 The number of save-game loads per mission

with high drop off rates did not have a correspondingly high number of save-game loads. In other words, players did not seem to be abandoning the game because the missions were too difficult. Three missions of the four that were set in the gang’s head quarters displayed this pattern of low failure rates and relatively high drop-off rates.

Now the hypothesis was that those four HQ missions, which had players completing tasks within a set time limit, were boring players and causing them to drop-off. But only having quantitative data to support this is not enough. Thus, we further tested this hypothesis with a follow-up qualitative community study that corroborated the HQ boredom hypothesis.

21.3.1.3 The Purpose: Development

In this case study, we used quantitative data analysis to develop a hypothesis as to why players were dropping off. The results generated new questions, which were further tested through different methods. The quantitative results showed the missions where players chose to stop playing. The results also showed which missions players persisted at by loading saved games multiple times. The picture of player behavior this quantitative data provided us was partial. We could not understand why players were behaving this way. Why did they persist in some missions of the game more than others? The ‘development’ of further research into these questions was informed by the results of the first, then the second quantitative analyses. We knew in which direction to dig deeper, and were able to target our inquiry to understand precisely why players abandoned where they did.

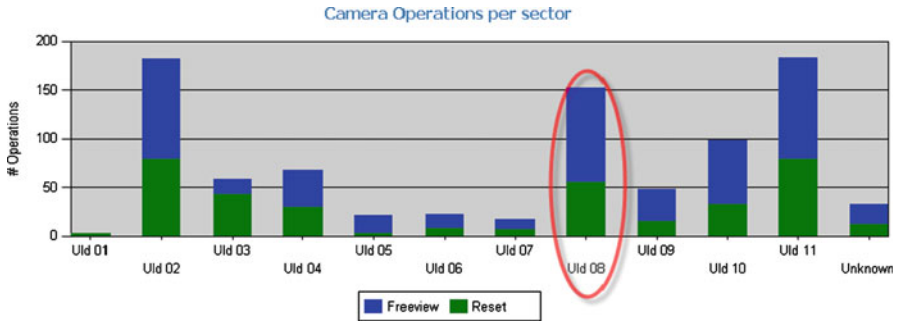


Fig. 21.4 Bar chart showing the aggregated number of camera operations performed by the players in a mission, broken down by sector

21.3.2 *Prince of Persia: Controlling the Camera on the Wii*

21.3.2.1 The Game

Prince of Persia: The Forgotten Sands (Wii) (Ubisoft, 2010) is a 3D platformer, which features acrobatics and agility. Throughout much of the game, the player must navigate through palaces and other environments by running across walls, ascending or descending chasms, jumping back and forth between walls, avoiding traps, climbing structures and jumping from platform to platform with well-timed leaps.

21.3.2.2 The Data and Analysis

The game is played from a third-person perspective, with unique controls to direct the camera using the Wiimote. A dedicated button on the Wiimote together with the motion sensing allows the player to manually control the direction of the camera. This raised the concern about the usability of this control scheme in that it might be cumbersome to the players. The game was instrumented to capture the positions on 3D maps where players used this camera control. We captured this and other data from play testers as they performed a walk-through of the game.

First, the raw counts of total camera operations were evaluated to see how many times players used the camera control (Fig. 21.4). Next, the data from play testers were analysed as in Fig. 21.5, showing precisely where, on a 3D visualization of the world, players used each of two types of manual camera controls on sector Uld 08.

The quantitative data provided a partial answer: Yes, the players were using the camera control often during their walk through. Yet what this data could not tell us is if this was truly a problem for the players.

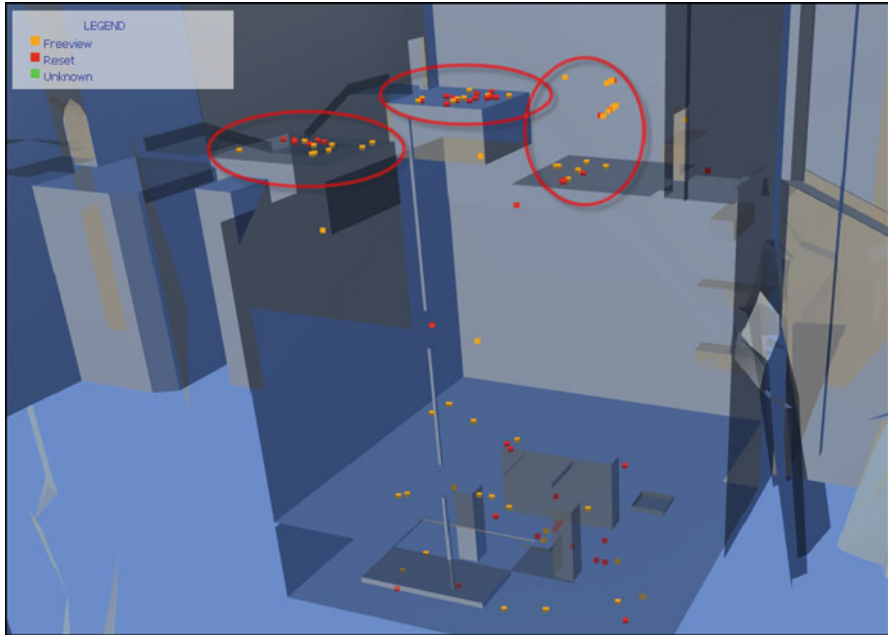


Fig. 21.5 Visualization of quantitative data: camera control use over the areas in the map

At the end of the playtest session, we asked players to fill out a questionnaire. A couple of these questions were specifically designed to probe the player's experience with the camera, and how comfortable they felt with the controls. While the quantitative telemetric results showed that players were making extensive use of the camera controls, analysis of the questionnaires revealed that the players did not mind the controls, and some players did not even notice how often they were manipulating the camera.

A side benefit to knowing where there are excessive manipulations of the camera is that we can consider alternative ways for players to reorient themselves, like scripted cameras as a player walks into a room, providing them a better sense of the space and the direction they should be heading.

21.3.2.3 The Purpose: Triangulation

In researching the usability of the camera controls, we purposefully cross-referenced qualitative and quantitative data. Together, these two sources provided a complete picture of the player's experience. Any uncertainty that existed from one method was clarified with results from the other.

If we had only quantitative data to measure the frequency of players using the manual camera controls, we could have been led to conclude that players were

overusing them, assuming that the controls interfered with their enjoyment of the game. On the other hand, with only qualitative data available, we could have argued that ‘they claim they don’t mind the control, but maybe it’s because they aren’t using it that much.’ By combining both quantitative and qualitative data, we could be more certain in our assessment that though use was high, players did not mind the controls.

21.3.3 *Crackdown: Agility Equals Fun*

21.3.3.1 The Game

Crackdown (Microsoft, 2007) is an open-world third-person role-playing game where players fight crime as a ‘super cop.’ The game was developed by Realtime Worlds and was tested at Microsoft Game Studios. The descriptions in this case study are taken from a talk given at the 2008 Game Developer Conference (Romero 2008).

To navigate the large 3D environment, players gain the ability to jump higher and higher as they progress through the game. When they perform certain actions, they gain points that can be used to increase their ‘agility’ allowing them to jump even higher and further.

21.3.3.2 The Data and Analysis

Late in the development process, the user research team at Microsoft described how they examined player ‘fun’ ratings in combination with the player’s progression in the RPG component of the game. Players were asked to rate the overall fun of the game, and also what they found most enjoyable with an open-ended question. The quantitative data collected during this same playtest measured how far each player progressed in the different RPG elements of the game, including their agility rating.

The user researchers at Microsoft analyzed the results at the end of the playtest session. In response to the qualitative ‘what was fun’ open-ended question, several players pointed to the ability to jump higher like a superhero. The combination of quantitative data (player levels in the RPG elements) with qualitative data (player fun ratings) showed that those players who gained the ability to jump onto rooftops with ease were the same players who were most likely to rate the game as fun. Furthermore, the open-ended question confirmed that the ability to jump higher was key to their enjoyment of the game. The analysis combined the qualitative fun ratings, the quantitative agility statistics and the development team’s design intention, that running and jumping as a super cop, was one of the game’s key features. The level design was adjusted to encourage players to level up their agility statistic more easily, and higher overall fun ratings followed.

Take-away lessons for this case study: the qualitative data (fun ratings) were used to isolate those players who had the most fun. Looking at the in-game behavior of those players, specifically how they leveled up their character's agility supported the theory that agility and fun were correlated. Questions about what made the game fun were answered using quantitative and qualitative results.

21.3.3.3 The Purpose: Complementarity

In this case study, results from quantitative data collection (those showing how much players leveled up in the RPG 'agility' statistic) were enhanced with the qualitative fun ratings of the play experience. The researchers were able to see a correlation between the agility statistic and the fun ratings.

21.3.4 Halo: Pacing and Player Progression

21.3.4.1 The Game

Another well-known example (Romero 2008) is Microsoft's Games User Research group's analysis of results from playtesting of Halo 3 (Microsoft, 2007). Halo 3 is a first-person shooter that has the player navigate environments on distant planets. Halo's missions are fast paced and designed to ensure players are continually engaged and experiencing action. When players get lost in the world, the pacing breaks down at which point there is a risk that players will stop playing.

21.3.4.2 The Data and Analysis

Data from the play experience were captured during playtest sessions using a telemetric system that collected quantitative player position data. This same system was also used to collect qualitative data about how players were feeling during their play experience. Every 3 minutes while playing Halo 3 the same survey would come up on screen asking players to evaluate their current feelings about the experience. The multiple choice answers included 'Too easy,' 'Not sure what to do,' 'Not sure where to go,' 'Too hard,' and 'I would quit.'

Figure 21.6 shows player positions when they answer the multiple choice survey. The quantitative data is the player's position on the map. The qualitative measure is the self-report data designed to gauge how they were feeling about their experience as they progressed through the mission. Together, the quantitative and qualitative data showed where players stated that they were frustrated (ready to give up) and how far along they were in terms of progressing through the mission. This method of mixing results enabled the research team to pinpoint difficult passages in the missions.

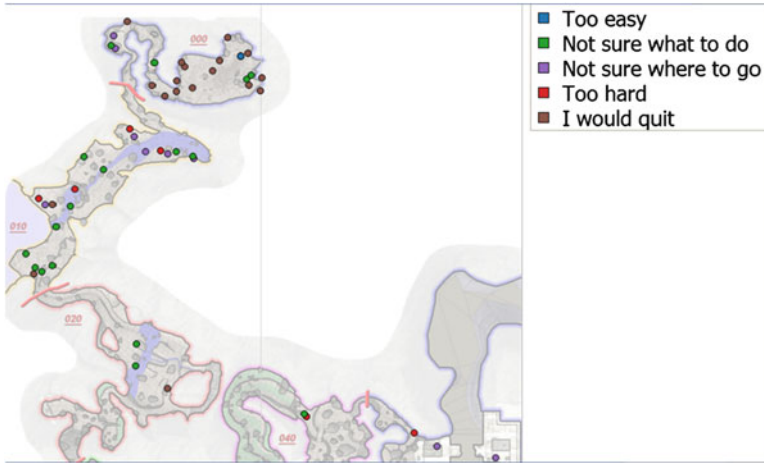


Fig. 21.6 Heatmap of self-report data and mission progression in Halo 3 (© Microsoft Studios; courtesy of Microsoft Studios)

21.3.4.3 The Purpose: Triangulation

To gain insight into player experience, the analysis tools allowed the researchers to triangulate quantitative information (where players were positioned on the map) and qualitative information (how they were feeling at that point). The researchers could provide detailed suggestions to the development team as to where to give more guidance to players to ensure a good flow within the level design.

21.3.5 *Assassin's Creed: Climbing faster*

21.3.5.1 The Game

Assassin's Creed 2 (Ubisoft, 2009) is the second game from the action-adventure Assassin's Creed series. These games are known as parkour type games featuring 'free running', climbing and jumping between rooftops in a city.

Assassin's Creed is an open world game that allows the player complete freedom in his choice of path to take when completing an objective. In contrast with linear games, where the boundaries restricting player movement are framed by corridor-like maps, a player's chosen path in an open-world game is restricted only by the boundaries of world, which can span many square kilometers. AC2 is played in various cities in fifteenth century Italy.

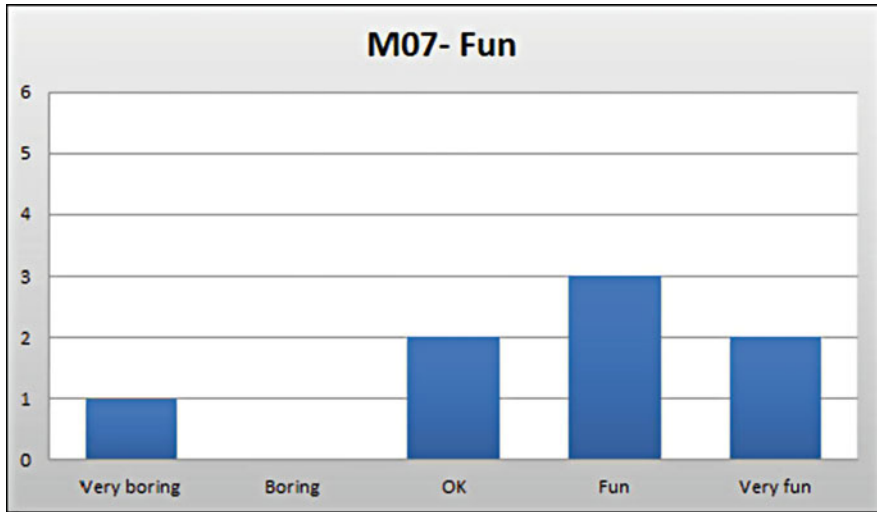


Fig. 21.7 Fun ratings for M07, test 1

21.3.5.2 The Data and Analysis

Early playtesting consisted of inviting eight players at a time to walk-through a selected sequence of missions from beginning to end. Upon completion, players were asked to rate each mission they had just played on a 5-point Likert scale. In one of the missions, players are required to locate and kill someone in three different parts of Venice, one of the cities reconstructed in AC2. Significantly, this mission is intended to take full advantage of the open world since players can achieve these objectives in any order, and get to the locations anyway they want. The intention of this kind of mission is for players to climb and fully use the free running features as they navigate to the different locations.

Looking at the 5-point scale, see Fig. 21.7, we can see that some players rated the mission as fun, while others did not. The testing methods also captured telemetric data on the player positions as they completed each mission. Figure 21.8 shows the heatmap for this mission showing a bird's eye view of the paths taken by the players through the city as they located and killed the three targets.

By triangulating paths data (quantitative) with this self-report qualitative data, the results revealed that those players rating the mission as either 'fun' or 'very fun' also used the roofs more than the other players. Although this finding confirmed that rooftops are where the fun is, it also generated new questions. The design team pondered what kept some players from using rooftops and free running as intended.

In this second opus of the Assassin's Creed franchise, the buildings in Venice were one to two stories taller than those in the first game, requiring more climbing time for the players to get to the roofs. The theory was that this additional time and effort to climb created an additional barrier, which discouraged the players from



Fig. 21.8 Player paths for mission 07, test 1

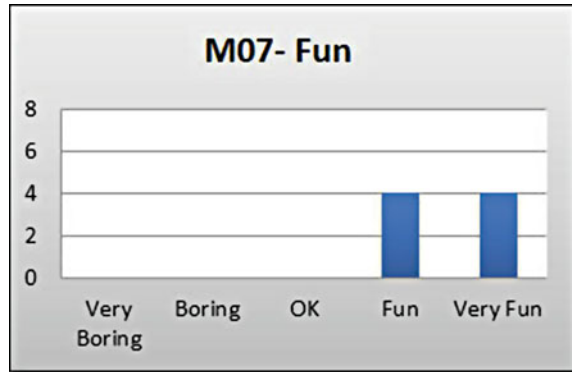


Fig. 21.9 Player paths for mission 07, test 4

reaching the rooftops. To remedy this problem, we modified the climbing feature, allowing players to reach the rooftops more quickly and easily, as discussed in Plourde (2010). The player could now reach rooftops twice as fast as what players experienced during early playtesting, thereby regaining the fluidity of the first title.

This change had the desired effect and the players employed the climbing and free running features as expected, seen in Fig. 21.9. Figure 21.10 shows the improved fun scores.

Fig. 21.10 Fun ratings for mission 07, test 4



21.3.5.3 The Purpose: Complementarity

The fun ratings showed inconsistent results between players. Some players had more fun than others. Quantitative in-game behavior (player’s chosen paths) was used to illustrate the differences between players who described the experience as fun and those who did not. In this case the researchers used the results of one method to clarify and understand the results from another method.

The Assassin’s Creed case is an example of the complementarity purpose since the results from the fun ratings needed the additional context of the path maps to give a complete picture of the phenomenon. This case also demonstrates the initiation purpose because interpretations of the qualitative data were inconclusive and required further exploring why players did not scale the walls and use the roofs.

21.4 Take Away Lessons

Games are tested at different stages of their development, often when all the features are still incomplete. Game designs change and evolve throughout the production process making it difficult to formulate specific rules or even to describe ‘best practices’ for researching videogames. However, there are some guidelines to keep in mind when starting out.

21.4.1 Focus on Key Features First

A game’s key features are those game designs which are intended to define the player experience with the game. Sometimes described as pillars or unique selling points, they are the various designs and mechanics supporting the main experience of playing the game. In Halo 3, a key feature is pacing, that is ensuring a good flow

in the player's progression. In Assassin's Creed, one of the key experiences designed for the player is running, jumping and climbing over vast environments. Ensuring that there are no barriers to fully experiencing these features is essential.

These features are usually described early in the conception phase of the game and it is around these features that research questions, data collection and analysis methods should first be designed. Since a large production team can be developing dozens of features simultaneously, it is impractical to attempt to research them all – at least in the beginning. The risk is that the research methodology, data collection and analysis are developed around something other than the designs that are meant to define the core player experience.

The benefit of identifying these features early in the game's development process is that research tools and methods will collect data that will likely have the greatest impact on quality of the game's experience (for more information on this process, see Chap. 13). This early focus can also help avoid expensive retooling late in the development process. Therefore it is recommended to focus early on what really matters: the core experience of the game and how it is intended to be delivered.

21.4.2 *Communicate*

Game development teams can vary in size from dozens (or less) to hundreds. On large productions, an additional challenge is to ensure that information reaches the right teams. The game design team needs to be involved in the user research process to provide the design intentions for the features. When telemetry is part of the data gathering strategy, programmers are necessarily part of this process (Fig. 21.11). Typically, there are two programmer roles: gameplay and tools. Gameplay programmers are the



Fig. 21.11 The communication triangle

ones who add the ‘hooks’ into the features, allowing for the game to log gameplay data. The role of the tools team is to process raw data into data structures that are ‘friendly’ for analysis.

Depending on the exact structure of the teams there are some key items that need to be communicated between the teams:

- The research team needs to understand the design intentions and the experience that players are expected to have as they interact with the various features.
- The programming teams need clear requirements regarding how the data will be collected and its intended use.
- User researchers are expected to provide feedback as reports to all the stakeholders.
- Changes and additions in data collection as the product develops and the research strategies evolve.
- Requirements and approvals need to flow through the triangle in all directions.

Together with key dates and milestones, continually circulate requirements and results to anyone involved in designing the features, collecting data and building the tools supporting the research.

21.4.3 Plan Out the Process of Inquiry

The user researcher’s first objective is to ensure that a game’s design intentions and key features are sufficiently well understood before starting to design the process of inquiry. It is a demanding and time-intensive task, nevertheless it is necessary to avoid the risk of investing considerable resources researching less crucial mechanics and features. Ultimately, the goal is to design the most appropriate collection and analysis strategy in the context of the project, its stage of development and the stakeholder needs:

- What methods of data collection are most appropriate to answer the research question?
- Are we too early in the development cycle to collect telemetric data? Should we first collect qualitative data and use those results to inform further strategies?
- Are the designs complex enough to justify collecting data from multiple sources and using a mixed-methods approach?
- Have new features been integrated into the game code, and can now be instrumented for telemetry?
- It is possible to set some resources aside for exploratory analyses (see Chap. 12)?

Some features are difficult, if not impossible, to measure through telemetry. If the objective is to determine what is frustrating players, it may be best to simply ask them to express how they feel. Qualitative methods and player observation can be deployed quickly and easily through paper surveys and checklists; the results can then be used to inform and shape follow-up research and methods.

While taking into account the current stage of the game's development, there are various methods available to measure a player's experience with a specific game design: open-ended questions, observation, telemetry data, video capture, focus groups or community survey, biometrics and more. However, not all are appropriate in every situation. Telemetry takes time to design, integrate into the game's code and test before any stable and informative data can be obtained. As features mature and change, the data gathering tools should be adjusted accordingly. Furthermore, new features can be researched as the game matures.

21.4.4 Correlation Is Not Causation

The existence of correlation between different statistics is not absolute proof of a relationship. This truism can still trip you up during analysis. If players are doing something often, like choosing most of the time to equip the pistol in a shooter game, and also rate their experience as fun, it does not mean that carrying the pistol is fun. By asking the players about their experience with the weapons we can get a more nuanced analysis of the feature. This is where the input of the game designers can be instrumental when interpreting results.

21.4.5 Integrate

As we have seen, player experience and behavior with a game can be monitored using different methods and for different purposes. The resulting data is interpreted and transformed into actionable information as part of the feedback loop from the user research department to the design team

In a playtesting environment we can simultaneously collect player data from different sources. When the results are integrated at the level of data analysis, we often see the most informative and relevant recommendations for the production team. In the Crackdown case study, the resulting data from the qualitative 'Rate your fun' and 'What was fun' questions was cross referenced with the quantitative level-up data to show that, as players leveled up, they also had more fun. In Assassin's Creed, players were grouped according to their evaluation of fun and the path analysis used this grouping to interpret differences in player perceptions and behavior.

21.4.6 Timing

Deadlines and code freezes are a reality in the game making business and the timing of results delivery is critical to the usefulness of the analysis. A highly

detailed report on player perceptions and behavior could be more or less actionable depending on the stage of the game's development when it is delivered. It is important to seek the right timing as well as the right data for delivering analyses and recommendations, if the development team should have a chance to act on them. Some research findings, no matter how detailed or insightful, will be only informative and not useful, if the development team can no longer make changes to the game.

21.5 Conclusion

Modern video games are interactive, truly multimedia experiences that must come together and remain coherent for the player for hours at a time. If the game is not compelling in some way, if it frustrates or does not resonate, the players will stop playing. User research studies are designed to gauge the quality of the play experience while capturing detailed in-game behavior data through an expanded repertoire of methods for collecting and analyzing game-play data.

When dealing with feelings such as fun, motivation and frustration, applying qualitative methods can give the proper context and enhance the interpretation of quantitative research findings. Mixed-method research combines what players think and feel (qualitative data) with what they actually do (quantitative data). Measures of satisfaction, challenge, and interest are linked to specific game play patterns, such as time spent, key mechanics used or paths taken in the virtual worlds.

Relying entirely on a single method of collecting player feedback can be insufficient to understand what motivated player behavior. As much as it is impossible for a human observer to record a player's precise path in the game's world, it is also impossible to use a quantitative instrument to truly understand why the experience was any fun. Though all methods have strengths and weaknesses, mixing methods is not about fixing the weaknesses, but rather amplifying the strengths of each to better understand the phenomena and to formulate the most relevant suggestions for improvements.

The case studies illustrate that when mixed-methods are applied, the whole is greater than the sum of the parts. In some cases, results from one method generated new questions, and stimulated new avenues for inquiry. This iterative approach requires a readiness to adjust the tools and research methods throughout the different stages of a game's production to ensure that conclusions and recommendations are always relevant, timely and actionable.

As games continue to evolve, presenting players with even more complex designs and moral choices, user research can benefit by employing advanced techniques for collecting and analyzing data from different sources. Mixing methods has the potential to yield a higher level of understanding of the player experience, generate more useful feedback to the development team, and ultimately make better games.

Author Bio

Eric Hazan joined the gaming industry and Ubisoft in 2005. He's been part of the team that built from the ground up a telemetry toolset and methods for gathering and reporting on gameplay data. He's worked on many Ubisoft titles including Assassin's Creed, Splinter Cell, Prince of Persia, Rainbow Six and Ghost Recon. He is currently the lead data analyst with the User Research team at Ubisoft Montreal. In the playtest lab, where participants are invited to take part in various usability tests, Eric works with the other analysts helping to turn raw data into useful information. Eric has a background in database design and programming. Prior to joining Ubisoft, he worked for many years as an IT consultant in customer relationship management systems (CRM) for a variety of companies where he was involved in all stages of projects, from initial conception and requirements gathering to training and support.

References

- Burke Johnson, R., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research, 1*(2), 112–133.
- Campbell, D. T., & Fiske, D. W. (1959). Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological Bulletin, 56*(2), 81–105.
- Collins, K., Onwuegbuzie, A. J., & Jiao, Q. G. (2007). A mixed methods investigation of mixed methods sampling designs in social and health science research. *Journal of Mixed Methods Research, 1*(3), 267–294.
- Davis, J. P., Steury, K., & Pagulayan, R. (2005). A survey method for assessing perceptions of a game: The consumer playtest in game design. *Game Studies 5*(1). Retrieved from http://www.gamestudies.org/0501/davis_steury_pagulayan/
- Drachen, A., & Sørensen, J. R. M. (2011, June 28–July 1). Arrgghh!!! Blending quantitative and qualitative methods to detect player frustration. In *Proceedings of foundation of digital games 2011*, Bordeaux, France.
- Greene, J. C., Caracelli, V. J., & Graham, W. F. (1989). Toward a conceptual framework for mixed-method evaluation designs. *Educational Evaluation and Policy Analysis, 11*(3), 255–274.
- Hopson, J. (2001). Behavioral game design. *Gamasutra*. Retrieved from http://www.gamasutra.com/view/feature/3085/behavioral_game_design.php
- Patton, M. Q. (1987). *How to use qualitative methods in evaluation*. Newbury Park: Sage Publications, Inc.
- Plourde, P. (2010). Designing assassin's creed 2. In *Game developers conference*.
- Romero, R. (2008). Successful instrumentation: Tracking attitudes and behaviors to improve games. In *Game developers conference*.
- Stern, H. (2011). Multivariate testing and game design: Methodology, metrics, and avoiding missteps. In *Game developers conference*.

Chapter 22

Combining Back-End Telemetry Data with Established User Testing Protocols: A Love Story

Jordan Lynn

Take Away Points:

1. Telemetry in digital games is an excellent supplement to existing qualitative research practices, and should be integrated into development, not just analyzed post release.
2. Heat maps and descriptive statistics can provide an immediate positive impact on the game's design.
3. Combining telemetry and qualitative research can minimize the drawbacks of each method used separately, and provides additional benefits beyond those of the individual methods.
4. Telemetry can provide a wide variety of unexpected benefits to different departments outside of user research.

As discussed in previous chapters in the book, many game studios are moving towards a model of collecting telemetry data of all players' actions within the game. Telemetry data analysis, as defined in Chap. 2, is the process of using in-game metrics to provide data about player behavior. This process is becoming extremely popular within the industry due to its power and utility in discovering “*who*” is performing “*what*” action “*when*” and “*where*” in your game. Previous chapters in this book described how such a process is initiated and investigated.

Unfortunately, also as discussed in various chapters of this book, even though telemetry analysis is a powerful technique, it cannot provide you with any reliable information about “*why*” a player is engaged in specific behaviors or not engaged in others, which, for a game that is currently in development, is the most important question.

J. Lynn (✉)

Volition, IncPlayer Experience Researcher, Champaign, IL, USA

e-mail: Jordan.Lynn@THQ.com

At Volition, we have recently developed a powerful telemetry software package, versatile enough to provide us with actionable data during game development and reliable enough to provide us with aggregate player metrics once the product goes live. In order to augment the objective data with subjective observations and player feedback, we combined telemetry analysis with existing user experience methodology. This hybrid quantitative/qualitative methodology will be the subject of this chapter. I will use case studies from my work at Volition to give practical examples and discuss results.

Even though some systems (such as Microsoft's TRUE system) exist that allow triangulation and visualization of telemetry data with qualitative data, these systems require large investments of time and money in order to develop and maintain. At Volition, we developed our telemetry system from scratch with little time and resources. The primary focus of this chapter is to describe how to implement a logging system and integrate it into your user testing practices if you lack the time and resources to create a comprehensive software suite.

The chapter is divided into the following sections: first, I will describe similar systems used within the videogames industry and the differences between those systems and the playtest/data analysis hybrid technique we employ at Volition. I will then briefly discuss Volition's telemetry system, list the most time-efficient processes we have identified, and discuss some of the more creative uses we have found thus far for our telemetry data analysis system. Following this discussion, I will explain several pitfalls that our research team identified through our trial-and-error process while integrating this methodology into our existing Player Experience testing schedule.

22.1 Microsoft's TRUE System: Description and Comparison to Volition Methodology

The premier proprietary system for analyzing the total player experience is Microsoft's "Tracking Real-Time User Experience" (TRUE) system. Kim et al. (2008) outlines the process used by Microsoft's research team to combine survey questionnaires, video recording, and real time metric analysis to gather a more complete picture of the playtesters' experience.

Researchers using the TRUE system recruit participants to play a game (or specific game segment) in development for a set period of time. The TRUE system is set to track User Initiated Events (UIE), events that occur when players directly interact with a system (Kim et al., p. 443). When a significant UIE occurs, the TRUE system records contextual data about the event in an XML file, attaches the associated video recording of the time before and after the event, and pauses the game to request additional input from the player. All of this data is then collated and compiled into a composite view that the researcher can then use to gather data about player experience.

The TRUE system is an integrated software and hardware suite that drastically improves a User Researcher's knowledge of the players' experience. However, not all researchers have access to this caliber of hardware/software integration, as well as the bandwidth and budget to create a proprietary user interface that condenses all of this complicated information in a simple-to-use package.

At Volition, we attempted to integrate several similar systems to those in the TRUE package in our local playtesting pipeline. In Sects. 22.2, 22.3, and 22.4, I will discuss the steps that Volition took to emulate and iterate upon this system. Additionally, I will discuss alternate beneficial implementations and outline difficulties and pitfalls of our implementation process in Sects. 22.5 and 22.6.

22.2 Volition's Telemetry System: Deciding What to Track

This chapter will be light on descriptions of the underlying technology. Instead, I will focus on the implementations that have proved to be the most beneficial to our design and development process. However, for reference, this section will briefly describe the function of our telemetry system. This is a very general discussion of a very detailed and intricate piece of software; please forgive me for any liberties I took to simplify the process.

The telemetry collection software developed by THQ is collectively referred to as the Games Data Service (GDS). Like many systems described in this book, the GDS software is based on a fundamental system of flagging significant events and recording information about those events in a database. Each time a significant event occurs (we predetermine which events we consider significant and wish to log), the system creates an entry in a table within the local database. As discussed in Chaps. 2 and 13, events range widely, from the gameplay, including player deaths, mission completions, to performance, such as framerate dropped, to general information, e.g., total number of players, players per platform, language selected. The data recorded into the table differs for each event, but each table provides detailed and useful information about the subject in question.

To give the reader a concrete example, I will use the player death event within Red Faction Armageddon (THQ, 2011). Each time a player dies in the game, we log a large set of data, which include the following fields:

1. A unique identification number for that specific death event.
2. Global status information, consisting of:
 - Player tag (each console is given a tag during playtesting to make correlating data and feedback easier).
 - Session_id, a unique identifier that allows us to correlate unrelated behaviors that occur within the same play session.
3. Player status information, consisting of:
 - Equipped weapon
 - Play time
 - Difficulty level
 - Current mission
 - Location (X, Y, Z coordinates)

4. Killer status information, consisting of:

- Killer name
- Killer's equipped weapon
- Killer's location (X, Y, Z coordinates)
- Fatal damage type (bullet, melee, explosion, Havok physics object, etc.)

Some events do not require as much data to be useful to the design team; for instance, on *Saints Row: the Third* (THQ, 2011), the following information is collected when a cheat is activated:

1. A unique identification number for that specific cheat_activated event
2. Global status information
 - Player tag
 - Session_id
3. Cheat Name

This data collection method allows us to reliably record “*who*” is performing “*what*” action “*when*” and “*where*”.

Additionally, as discussed in Chap. 3 and in other chapters in this book, there are many stakeholders who are interested in this information. At Volition, the stakeholders include:

- System Designers, who are primarily interested in gameplay and balance related issues during prerelease testing and player preference data for franchise development,
- Level/Combat Designers, who are interested in navigation and combat balance issues,
- Producers, who are primarily interested in feature usage (for scope control)
- Technical Artists, who are interested in game performance and system behaviors,
- Quality Assurance, who are interested in tracking testers and automating QA testing tasks for increased efficiency.

Determining which items to log and what data should be recorded for each significant event is the subject of several meetings between the User Experience practitioners, the design team, and the production team. Each stakeholder is expected to understand how the system works and develop realistic expectations for how the data will be applied.

22.3 Volition's Telemetry Data Analysis: Efficient and Concise Methods

Between extensive playtesting and a large commercial release, even an efficient data tracking system can record incredibly large amounts of data. When faced with a massive database filled with gigabytes of player data in hundreds of tables, analysis

paralysis can set in and severely delay the transmission of useful data to the design team. A delay of even 2 weeks between testing and analysis can invalidate a large portion of the results due to the constantly shifting nature of game development. Thus, we need an efficient process whereby we can gather our findings, organize them logically, and ship them out to the team as rapidly as possible to maximize the positive impact of the data analysis process.

Additionally, the report also needs to be compact and efficient in its use of visuals and words. Designers and producers face strict deadlines to various tasks, including fix bugs, develop systems, balance existing systems, elicit and implement feedback – often, they simply do not have the time to read a nine page report detailing all of the problems with a certain mission. Concise descriptions of issues, detailed graphs, and short highlight video clips of player behavior cost you less time to create and cost the designer less time to ingest.

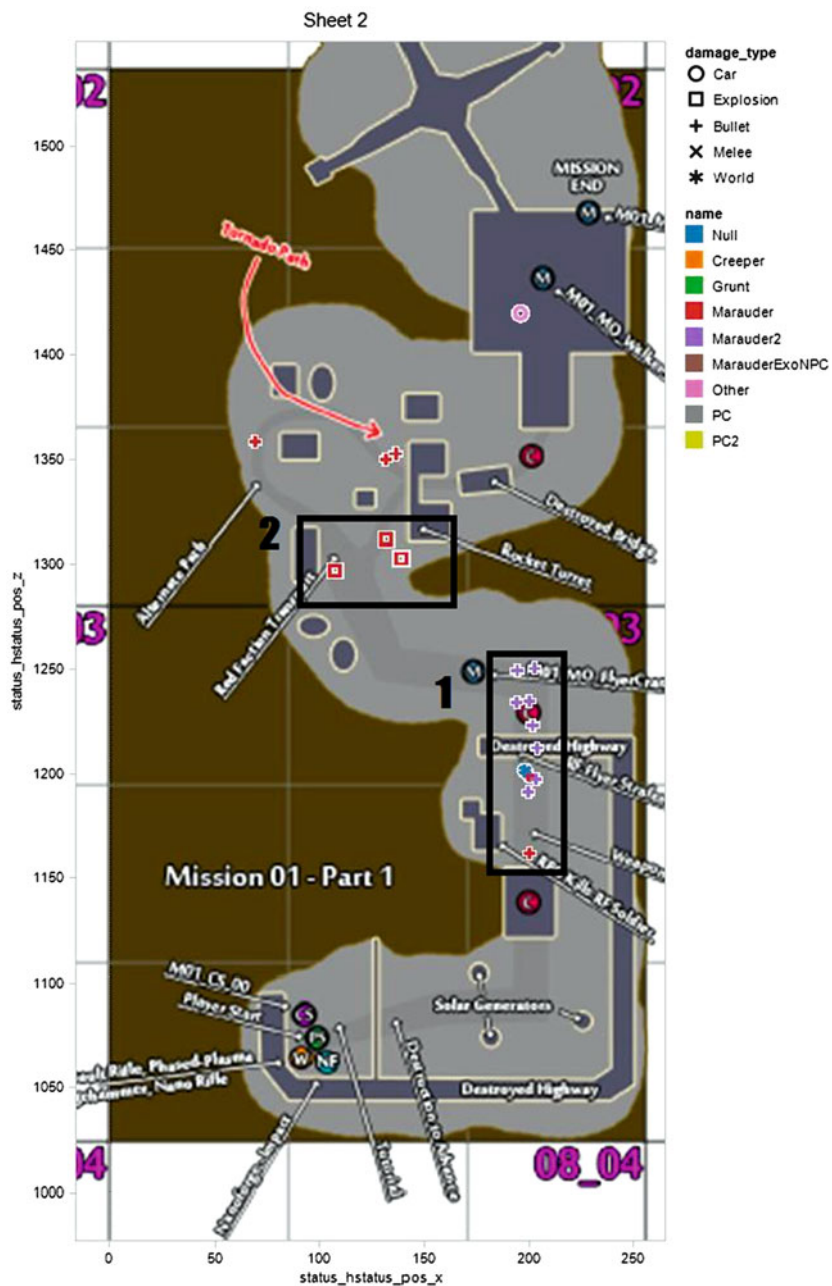
Within these constraints, we have discovered several methods that provide us with the best and most efficient techniques for rapidly creating reports that are concise, informative and impactful. I will discuss these methods here.

22.3.1 *Heat Maps*

Heat maps are data overlaid onto a map of the game space. Such heat maps can be created by many applications, such as the Geographic Information Systems (for a more detailed description, see Drachen and Canossa 2009, pp. 182–189, or see Chap. 16). The most useful heat map Volition used on *Red Faction Armageddon* was the player death heat map, shown in Fig. 22.1. We used Tableau to plot the players' death location on the X and Z axes overlaid on top of the game's world map. This method allowed us to quickly and easily show our level designers exactly where clusters of player deaths were occurring. These maps took very little time to create and provided very concrete data on the difficulty balance of scripted combat encounters. If a designer expects only one or two deaths in a certain mission, and instead sees 14, most of which occur in the same corridor, the designer can then understand the exact nature of the problem, the exact location of the issue, and thus, make changes accordingly to the design. In the following sections, I will discuss more concrete examples of this dialogue between the user experience team and the design team.

Figure 22.1 shows an example heat map of each player's death in the first mission of *Red Faction Armageddon* during one of our initial playtests. Each symbol represents a unique player death. The color of the symbol denotes the identity of the character that inflicted the lethal damage. The shape of the symbol denotes the type/source of damage.

Location 1 indicates the most pressing issue revealed in this playtest. This corridor is the first combat encounter of the game, and the design intention is for players to learn the basic mechanics of combat by engaging enemies in a constructed corridor. However, 11 deaths were recorded by our telemetry system (for only 8 players), indicating that our first combat encounter was inflicting an alarmingly high



Status_hstatus_pos_x vs. status_hstatus_pos_z. Color shows details about name. Shape shows details about damage_type. Details are shown for killer_type. The data is filtered on start, which includes dates on or after 10/13/2010 8:38:55 PM.

Fig. 22.1 Player death in mission 01

number of casualties, including one player death (the cross at the bottom of the outlined area) that is only 5 m into the combat encounter. After showing this diagram to the mission designer, the difficulty was adjusted, enemy type and placement were tweaked. As a consequence, the number of player casualties in the corridor was reduced to virtually 0 in future playtests.

Location 2 contains only three deaths, which is not an unusually high concentration for this portion of the level. However, the square denotes an explosive kill from a particular enemy, in this instance, an NPC on a rocket launcher turret. This NPC was not intended to pose a significant challenge to players; the three deaths depicted here (as well as additional deaths in other playtests) point to an unintentionally high number of casualties in this encounter. Upon further investigation during subsequent playtests, we discovered that players were not targeting this opponent since the turret is located 45° above the player's line of sight, and the game did not provide enough prompting to the player to look up and target the threat. After improving the signaling provided by the level design, players identified the threat more consistently and died with much lower frequency in subsequent playtests.

Using heat maps produced spectacularly useful data for our level and system designers. However, there is a question of what information to display using these heat maps. In our experience, some heat maps produced a good visual tool for the dialogue between the user research team and the design team. Below, I discuss some of the types of heat maps that we found useful.

22.3.1.1 Heat Maps Displaying Player Death

These maps visually display areas with concentrations of player failures, allowing designers to see if the difficulty balance of a level meets their expectations. Using color to code the difficulty level, we could easily highlight areas where casual, normal, and hardcore players were having trouble; we once caught a major difficulty spike on casual difficulty using one of these heat maps. For example, one particular encounter required players to shoot a suspended object, dropping a bomb onto a boss. The heat map for one play test highlighted the fact that this specific encounter caused dozens of player deaths, while the area was expected to cause zero player deaths. Upon further investigation, we discovered that players took multiple attempts to realize that the objective was to shoot the suspended object. After a quick change by the level designer, in the next play test there were no deaths in that specific location.

22.3.1.2 Heat Maps Displaying Out of Ammo

These maps display areas where players run out of ammunition for their equipped weapon. These maps allow designers to accurately predict where to place sources of ammo for optimum ammo balance. In *Red Faction Armageddon*, weapons are the players' primary method of interacting with the world; running out of ammo hinders the player and causes frustration. These heat maps highlighted a major issue with

players bypassing ammo pickups, which we then tested using traditional play test methodology as described in Sect. 22.4.1.

22.3.1.3 Heat Maps Displaying GPS Request

These maps display areas where players activate the in-game GPS navigation system (with the press of a specific button, arrows appear on-screen indicating the direction of the next objective). Mapping the uses of the GPS was an excellent way to discover where players were getting lost. These maps were doubly important since many of *Red Faction Armageddon*'s environments are dark caverns with multiple branching paths. By analyzing these maps we could identify areas where an abnormally high number of players were activating the GPS, allowing our level designers to improve mission flow and navigation between play tests.

22.3.1.4 Heat Maps Displaying Player Breadcrumbs

Every X seconds (we generally set X to 10 or 30), the game submits the player's status to the database (location, health, equipped weapon, etc.). These breadcrumbs can be plotted on a map to display the rate at which players physically progress through the game. This is similar to what was visualized and discussed in Chap. 19, and in the works of Moura et al. (2011). These maps can be useful for finding areas where players are spending too much time, possibly because of navigation difficulties. Conversely, if there are "showpiece" areas where we want the player to spend extra time interacting with a game element, these breadcrumbs can tell us if players are failing to do so. Additionally, for several tests, I plotted the breadcrumbs with color coding according to the amount of health the player had at that moment; this allowed me to diagram a map of every place players had full health, moderate health, or were near death.

22.3.2 *Simple Tables and Charts*

Several data points that we discovered did not require complex visualizations. A simple graph or report of the relative values can assist designers in balance passes and level iteration. Below I describe such examples.

22.3.2.1 NPC Death Event

Similar to player death, this event records every death of a Non-Player Character (NPC), what kind of NPC it is, location, killer weapon, and fatal damage type. Using this data, we can gather very interesting data about how players interact with the NPCs

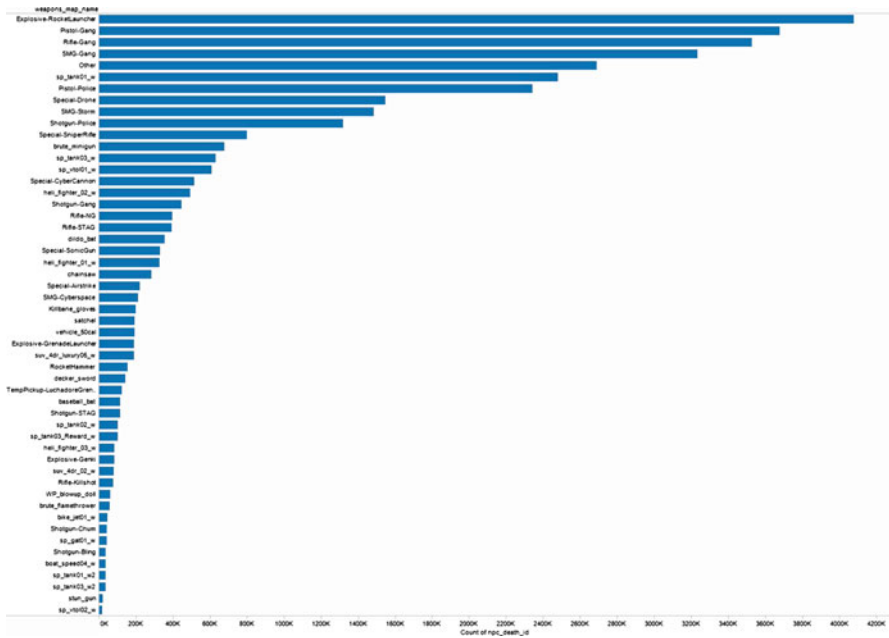


Fig. 22.2 Weapon’s use graph

in our games. This event was extremely useful in balancing our weapon loadouts. We ran a large number of participants through our game; using this table we could identify which weapons players were using, how many enemies they were killing, and which types of enemies they killed with which weapon. As illustrated in Fig. 22.2, we discovered that certain weapons were far more commonly used than others; after designers tweaked the weapon balance, the weapon usage statistics normalized so that no one weapon was the obvious weapon of choice for all players.

This event also allowed us to analyze when and where players were using specific weapons. Several of our weapons are showpiece weapons that designers expect players to utilize immediately, preferably for a “wow” moment. Through telemetry, we were able to identify areas where these weapons were being used and, more importantly, where they were being ignored. We also found some interesting situations where enemy NPCs were being killed by other enemy NPCs. This resulted in the discovery of multiple bugs, and although not directly related to player experience research, correcting these issues caused an improvement of the overall quality of our product.

22.3.2.2 Player Death Event

Even without heat maps, this event provided useful information. We could easily highlight the details of player failure conditions as the game progressed, which allowed designers to see if the difficulty was scaling properly throughout the entire

Table 22.1 Player deaths by cause

Name	PC death by cause						Grand total
	globals_tag						
	615station1	615station2	615station3	615station4	615station6	615station7	
Other	19	3	8	15	4	20	69
Special-SniperRifle	4		1	14	3	2	24
Pistol-Gang	4		1	11	7	1	24
heli_fighter_02_w	3		2	7	1	8	21
brute_minigun	1	2		3	5	5	16
TempPickup-LuchadoreGren.	1	2	2	2	2	6	15
SMG-Cyberspace			4		4	2	10
SMG-Gang	3			5			8
suv_4dr_02_w			2	2	2		6
Shotgun-Gang			2	3		1	6
RocketHammer	2		2	1	1		6
sp_vto101_w			2		1	2	5
Pistol-Police	2			2	1		5
Avatar_sword					4		4
suv_4dr_luxury05_w	3						3
Shotgun-Police	2		1				3
sp_tank03_w			2				2
sp_gat01_w	1		1				2
SMG-Storm				2			2
Shotgun-STAG	2						2
RocketHammer_VR					2		2
Rifle-STAG	1		1				2
stun_gun	1						1
sp_vto102_w						1	1
sp_tank01_w2	1						1
sp_tank01_w			1				1
Rifle-Gang	1						1
Grand total	51	7	32	67	37	48	242

gameplay experience. As illustrated in Table 22.1, during this playtest for *Saints Row: the Third* the Sniper Rifle accounted for a disproportionately high number of kills against the players, especially the player at Station 4.

Aggregate numbers can illustrate which enemy weapons may be over- or under-powered. Looking at the data for several entire play tests revealed which enemy weapons were causing a disproportionately high (Sniper Rifle, Gang Helicopter, and Rocket Hammer) or unexpectedly low (Grenade Launcher) number of player deaths.

These tables of data allowed us to provide detailed and useful information with a moderate amount of effort. I highly recommend these implementations for any project, as they are especially useful in assisting design teams and verifying designer expectations.

22.4 Volition Triangulation Techniques: Synergy Between Existing Research Practices – Discussions on Process

22.4.1 Existing User Research Practices

The traditional testing practices at Volition are survey and observation based testing methods (Isbister and Schaffer 2008). We conduct “think aloud” and interview based tests, but the testing methods that benefit most from telemetry analysis are the larger, survey and observation based tests, as these tests use larger samples over longer time periods. As such we will focus primarily on those kinds of tests for this discussion.

For any readers unfamiliar with survey based testing, I will provide a brief description of our methodology. Near the middle of the production phase of game development, we begin recruiting participants for tests that last from 4 h to an entire week, depending on the functionality and stability of the game. Players come into our local office and are instructed to interact with a specific portion of the game; in later tests, they may be instructed to simply play the entire game from start to finish. Players are provided with two monitors; one monitor displays the game, and the other displays a survey. Once players reach certain milestones in the game (usually the completion of a mission or activity), they are instructed to provide feedback and ratings for the content they have just experienced.

At the same time, the developers and researcher observe the players’ in-game behavior and performance via instant streaming of a live video feed from each player’s workstation. Developers then record any observations noting aberrant behavior, unexpected results, or any areas that appear to cause frustration, confusion, or any other unintended negative responses.

Additionally, each playsession is recorded and timestamped to allow for retroactive analysis of in-game behavior. These recordings are especially useful in creating highlight reels of significant events across multiple participants, and allow designers to understand the context surrounding these significant events.

Once the test is completed, the researcher collates the player feedback, the developers’ observations, and the researchers’ observations into a prioritized document that is sent to the design leads for verification. Tasks are then assigned to team members to address the issues revealed in the test. Before the addition of telemetry, the vast majority of this feedback was subjective; once metrics were integrated, several problems, caused by the subjectivity of the data, were greatly alleviated.

22.4.2 *Triangulating Users' Comments with Evidence Through Telemetry*

Everyone in the field of user research has had to address the problem of participant ambiguity at some point. Players are not always able to specifically identify the issues that they are facing, and may simply provide generalized comments. “This level is too hard” seems straightforward, but if the researcher is unable to follow up with the participant, how does he or she deduce the exact source of this comment? Is the enemy AI too aggressive or numerous? Is the ally AI ineffective? Is there a level design issue creating a bottleneck or other disadvantageous combat scenario? Did the player get lost, and did the difficulty arise from navigation issues? Thankfully, telemetry can assist researchers in identifying the low-level specifics of such comments, which allows designers to take direct and impactful action.

Using “This level is too hard” as a specific example, here are a few ways to examine the issue more thoroughly:

- First, check the mission failure logs. These logs can show the type of failure. Did the player die, let an important NPC die, fail to complete a timed objective quickly enough, or fail to achieve the required score? Did this specific player fail in a specific way multiple times? This type of log can also reveal patterns across participants. Did all players fail in the same way at least once, but this player failed more times? Finding a consistent pattern could indicate a problem.
- Second, delve more deeply into the specific failure conditions. Assuming player death as the primary failure condition, check the player death table to examine the reasons for the player’s deaths. Look for patterns; did a specific enemy or weapon cause a disproportionate amount of deaths? Did the deaths occur in the same (or very close) locations? Alternatively, assuming there were no failure conditions, one could check the breadcrumbs log to follow a player’s route through the level; does he or she seem to be spending an inordinate amount of time in one area? Is it possible that the player is unable to navigate through that segment? Finding a pattern in the logging can present a designer with very useful feedback on how to address the participant’s difficulties.
- Finally, check ancillary logs. For mission difficulty, I always check the logs of NPC deaths. Calculate the total number of enemies the participant fought in combat; does this number match the designer’s expectations? Check the locations of failure conditions across participants; are they all failing at the same spot? Check the ammunition logs; are the players facing a scarcity of ammo for their preferred weapons?

Investigating the data logs can provide some interesting insights into the specific causes of general complaints. Analyzing the data this deeply may require more time, but the return on investment is typically very high.

22.4.3 Integrating Telemetry with Existing Practices

In the last section, I discussed an example of using telemetry analysis to resolve and clarify ambiguous player feedback after a test is complete. However, incorporating observations from telemetry during the middle of a play test is a little trickier, but can be very useful. I will discuss below some examples to show how we were able to integrate these two methods together.

22.4.3.1 Adding Questions to Survey

By monitoring the data in real time, it is possible to find issues and anomalies that require additional information to understand properly. During long term play tests, especially those that stretch over multiple days, it is possible to adjust surveys to ask for clarification on issues discovered through telemetry. For example, in a recent play test I conducted, the telemetry showed that grenades accounted for a disproportionately low number of combat kills. On further investigation, none of the players in the playtests had acquired grenades through the weapon store interface. Did players understand how to acquire grenades, but chose not to, or did players fail to realize how to acquire grenades? In this instance, by monitoring the data logs, we were able to adjust the following day's survey to ask participants about acquiring and using grenades.

22.4.3.2 Locating Areas of Concern, and Asking Designers to Tune in

The ability to capture data and report it in real time has another added benefit; designers can be alerted when issues are arising in their particular areas of responsibility, and can observe the issues themselves as they occur. In Volition's existing playtest program, designers are able to stream the video from each play test participant directly to their work station; however, most designers are too busy to focus exclusively on watching the video streams for a protracted period of time. By monitoring the data logs, the test administrator or a designated observer can note anomalous data, alert the relevant designer, and have that designer direct his attention toward the potentially troublesome issue as it happens in real time.

For instance, if the data logs reveal that one player died multiple times at the same location, an observer watching the logs can inform the mission designer and the combat designer of the situation and direct their attention to the appropriate video stream. By direct observation of a potential issue, designers can understand the full context of the event. This provides significantly more information than a report after the fact.

Additionally, with enough time and resources available, this process can be automated. An automated script can tally counts of specific events (player deaths, out of ammo events, mission failures, just to name a few) and alert the test moni-

tor to anomalous data in real time. Depending on the complexity of the tools available, the script can directly notify relevant designers or other persons of interest as well.

22.4.3.3 Creating Highlight Reels Using Time Stamps

Additionally, if perhaps the designers are unlucky enough to miss a repetition of the event in question, the data logs contain a time stamp for the event, which allows a researcher to isolate the event in the post-test video recording and send the visual record directly to the relevant designers. These highlight reels are extremely helpful in illustrating observations about player behavior and impart far more information than a standard report. Therefore, implementing such a system into your existing play test practices can augment the amount of information provided by each play test. Implementing those solutions in real time increases those gains even further.

22.5 Incorporating Telemetry into Areas Outside of Play Testing

The telemetry system can also be extremely useful in applications outside of play testing. Here are a few of the ways that Volition has adapted its telemetry system to provide information in other arenas.

22.5.1 Bug Tracking

Since the current telemetry system is optimized for evaluating the users' experience, bug tracking is not the most direct application of the data. However, certain observations can lead to the discovery of bugs that the team can then eliminate. Two of the most useful observations are about "friendly fire" incidents among NPCs and examining frame-rate dips.

In *Red Faction Armageddon*, enemy NPCs cannot damage one another with their weapons. According to the data logs, however, a significant number of NPCs were dying with the killer being identified as another enemy NPC. Thinking this was a bug in the telemetry system itself, we began investigating and attempting to recreate the data. As it turns out, the enemies could not damage each other with weapons, but they could jump and collide with one another, causing a significant amount of damage to both parties. In dense combat areas, enemies were stomping each other to death. This bug was fixed, significantly reducing the number of NPC-attributed kills.

Additionally, a custom event to determine certain performance statistics can help identify areas where performance can be optimized. A good example is adding a

measurement for frames per second to the player breadcrumb data table. Every 10 s, the game records the player's location, his status, and the game's current frame-rate. Examining this data using playtest participants can highlight areas with alarmingly low frame-rates; however, enabling this data log for every build of the game, so that every designer, QA tester, programmer, and artist are all gathering back end data, will almost certainly identify the vast majority of areas in the game with unacceptable frame-rates.

22.5.2 Quality Assurance (QA) Productivity Tool

My interaction with the QA department led to the discovery of alternate applications of telemetry system in the daily operations of QA. Within a few short weeks, data mining was implemented as a productivity monitor for QA (similar to what is used by Bioware, discussed in Chap. 7); each specific QA tester was assigned a specific user ID in the telemetry tables, so that their specific data can be isolated. Each week, the QA leads can load the relevant data logs to see exactly what missions, weapons, situations, and locations were tested by each individual member of QA. This allows the leads to monitor productivity efficiently and to make recommendations for improvement for specific individuals.

In addition, this system also provides an excellent picture of which areas of the game are *not* receiving adequate QA coverage. The QA leads were initially worried about verifying that every area of the game received QA tester coverage; with the support of the telemetry system and a dedicated QA data server, the leads were able to investigate the thoroughness of their coverage and make adjustments from week to week. For instance, telemetry data visualizations can easily display the number of kills of each enemy type by each weapon; were the QA testers using every weapon on every enemy, or were they subconsciously sticking with their personal preferred weapon loadout? This is only one example of the many ways that QA was able to benefit from the telemetry system.

22.6 Thoughts on Breakdowns and Pitfalls of Integrating a Telemetry System into the User Experience Pipeline

While the telemetry system is powerful and extremely useful, there are several pitfalls to avoid when implementing such a system. Integrating a new tool into an existing user research program can be extremely valuable, but will not come without hiccups, miscommunications, and the occasional error. Hopefully, the following list of issues I have dealt with over the last year can help interested readers to avoid similar pitfalls and setbacks and increase the speed of successful implementation.

22.6.1 Analysis Paralysis: How Much Data Is Too Much?

The very first thing that should be considered before implementing a telemetry system is how the data will be used. It hardly matters if a game logs every single activity that a player can undertake in the entire game, if there is no thought put into how that data will actually be applied to improving the game itself. Resist the temptation to record vast quantities of data just because such a thing is possible. Sit down each of the relevant stakeholders in the process: design, production, marketing, QA, and the programmer(s) who will be tasked with the actual implementation. Ask each person to produce questions that they would like answered, no matter how specific or abstract; once that list is compiled, work with the programmer(s) to discover the most efficient method for answering those questions through telemetry.

22.6.2 Understanding Limitations: Does Everyone Have Realistic Expectations for the Uses of This Data?

Continuing along that same line of reasoning, make sure that each stakeholder understands how this data can, and cannot, be used. Questions about player motivation cannot be adequately answered by a data log; those questions are best left to a mixed-methods approach, combining player feedback (interviews and surveys) with the quantitative component of telemetry analysis. Data logs can provide a large quantity of data to describe a situation, but they cannot report player opinions, likes, dislikes, frustration, or confusion. Correlating findings from telemetry with player observations almost always improves the quality and accuracy of the analysis.

Also, make sure that the team does not overreact to telemetry results. A data log that reports an excessive number of player deaths in one location may seem troubling and could lead a level designer to quickly alter the layout or combat scenarios of the area; however, always check the player's feedback. Some areas have looked absolutely dismal on the heat maps with extreme numbers of player deaths, while the players report thoroughly enjoying the same experience. In the end, the player's enjoyment of the game and the meeting of designer expectations are the most important goals.

22.6.3 Development and Implementation: Who Is Responsible for Tweaking and Updating the System?

Establish the workflow for correcting and tweaking the telemetry system early in the process. Like any other system, the telemetry system is going to occasionally break down or encounter errors in the game code. Depending on the severity of these issues, as well as the level of dependence on the quantitative data they affect, these issues may need to be corrected quickly. Is there enough bandwidth to have an

engineer dedicated specifically to the telemetry system? Most programmers have large demands on their time, especially as projects near completion (which happens to be when play testing is most frequent). Make sure that there is a clear chain of contact for the inevitable issues that will crop up.

On a related note, secure time for the QA department to investigate the telemetry system to verify that implemented systems are working as intended. Ensure that QA has access to adequate tools to verify events that are occurring in game are accurately being logged in the database; this may include supplying QA with data visualization software, which can be costly and may take time to be arranged properly.

22.6.4 Reporting Pipeline: Where Is the Data Going?

Before telemetry becomes an integral part of your testing regimen, determine exactly how the data will be used. What is the most effective way to present the relevant data? Large comprehensive reports delivered to the leads and producers? Summaries and the most relevant statistics sent out in a bullet-pointed list to the entire development team? Or simply as a supplement to existing observations? In an established user testing program, there are probably previously established pipelines for the flow of test information, but telemetry provides a wealth of useful data that can be more granular than most play test reports. Consider finding new ways to provide useful information to individual team members; sending a heat map of player deaths to each mission designer, for instance, can help that designer determine if the player experience in his or her mission matches expectations. Each project will be different, so consider how you can make the most significant positive impact on the project through your reporting of the telemetry analysis results.

22.7 Final Thoughts

Telemetry is a powerful and incredibly useful tool for a user researcher, and an excellent addition to any researcher's repertoire. However, like in many areas of research, the most efficient way to study any human experience cannot be simply quantitative, or rely solely on qualitative methodologies. A mixed-methods approach, one that blends the strengths of both powerful technology and the investigation of participants' internal motivations, can provide a robust depiction of the state of a game in development and provide invaluable feedback to the designers and producers working to ensure the quality of that title. Even without the benefit of an integrated software package, like the TRUE system, a user researcher can begin producing relevant and insightful reports backed by comprehensive data logs, increasing the accuracy of your reports and improving the overall quality of the title.

About the Author

Jordan Lynn is currently employed at Volition, Inc., performing Usability and User Experience studies for *Red Faction: Armageddon*, *Saints Row: the Third*, *Saints Row: the Third: Enter the Dominatrix*, and additional unannounced titles. He has an MA in Journalism and Mass Communication from the Grady College at the University of Georgia with a Graduate Certification in Media Industry Research.

References

- Drachen, A., & Canossa, A. (2009). Analyzing spatial user behavior in computer games using geographic information systems. In *Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era – MindTrek 2009* (pp. 182–189), Tampere.
- Isbister, K., & Schaffer, N. (2008). *Game usability: Advancing the player experience*. San Francisco, CA, USA: Morgan Kaufman.
- Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B. C., Pagulayan, R. J., & Wixon, D. (2008). Tracking Real-Time User Experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceedings of CHI 2008* (pp. 443–451), Florence.
- Moura, D., Seif El-Nasr, M., & Shaw, C. (2011). Visualizing and understanding players' behavior in video games: Discovering patterns and supporting aggregation and comparison. In *Siggraph 2011*, ACM: New York.

Chapter 23

Game Metrics Through Questionnaires

Ben Weedon

Take Away Points:

- Questionnaires are a quick, inexpensive, low-tech means of benchmarking user reaction to games.
- Questionnaires can give you valuable data on how gradual, subtle changes to the game over different builds affect the player's appreciation of it as it develops.
- Testing your questionnaires in small trials before the main tests (pilot sessions) are essential to ensure that your measures are providing accurate and useful data.
- It's important to try out different measures as part of evaluating your game. Experimentation is important.
- Quantitative benchmarks are useful for developers to eyeball the headline comparisons between rounds of testing, but be sure to obtain the qualitative feedback too, so you can understand the reasons behind the scores.

23.1 Introduction

Our goal as user researchers is to provide the developers and producers with the most informative insights into player attitudes, expectations and reasons for play, helping them make decisions to enhance their game designs. The term 'enhance' could be defined in several ways: more fun, more aligned with the designers' intentions, or maybe more likely to appeal more to the core audience. The definition of the outcome criteria and goals of the research is essential to clarify early when engaging in game user research.

B. Weedon (✉)
PlayableGames, London, UK
e-mail: Ben.weedon@serco.com

The company I work for, and am writing on behalf of, ExperienceLab, is a user research consultancy. When clients ask us to work with them we always first discuss the outcome criteria and goals of the research, to help us understand what the client thinks will enhance their game.

Whilst as game user researchers we are always looking for new methods to make the user testing process as efficient as possible, to provide accurate and useful insights, we also – as external consultants to a client – need to consider the time and expense of investing in and implementing any method, and trade these off against their benefits to the research. We need to think about how to make the most of the time we have with the title we are researching. This is important from a cost and time perspective, as developers need good quality information on their game as quickly as possible, so that they can start to act on it and make changes. Obtaining results from some methods, e.g., biophysical, can sometimes take too long, due to the post processing and analysis involved, but other methods, such as questionnaires, can generate information almost immediately, and provide an inexpensive overview of the current game status. Taking a snapshot of the game through participants' responses to questions allows us to establish a benchmark of how the game is rated by participants at that particular stage of its development.

Benchmarking is an important way of understanding how changes in the game's design are affecting the gameplay experience. At ExperienceLab we recommend its use whenever possible. In this chapter I will outline the benefits of questionnaires as user research methods, and how to derive quantitative measures based on these. This takes a different approach than most of the chapters in this book, which are focused on telemetry-derived metrics. In contrast with telemetry systems, surveys are a low-tech and inexpensive way of obtaining insights into player behavior, and also form a great supplement to telemetry-driven metrics.

23.2 Understanding the User Research Goals

Often clients (game companies) have an idea of what they want us to investigate. For example, there is usually the need to understand if the gameplay is interesting and fun, whether the difficulty curve is appropriate to the audience, or how the controls work, but, there are often other specific questions relevant to a particular game or level, or part of a level. For example, one of our clients wanted to know how users would react if they varied the speed of character movement ever so slightly in different tests. They wanted to optimize the character speed so that players could move swiftly around the game world, but maintain a level of movement accuracy so that they could easily interact with elements without whizzing past them. The clients are the experts on the game, since they conceived it, and so they often know what research questions they want answered.

Situations where the client has a request for a specific methodology are less common. It is usually up to us as user research consultants to decide which methodology

works best to meet the research goals, even though it is the client's final call as to the exact approach to use. The rest of this chapter describes when it is best to use questionnaires in relation with user testing, and also when it is not so useful.

23.3 Questionnaires and Why We Use Them

Questionnaires are valuable as a tool to the researcher for a variety of reasons. They are quick for participants to complete (or they should be), and often quick to analyze once the data has been entered into a spreadsheet.

Where time is short during sessions, we need to gather as much data as we can, as efficiently as possible. This means pen and paper is our preferred method, even though it means we have to enter the paper-based questionnaire data ourselves at the end of the session. Also, we always keep the original paper responses to refer back to, in case the electronic data in the spreadsheet or online gets corrupted.

We tried using online survey tools during lab sessions, such as SurveyMonkey, to host our questionnaires online and collate the data. However, we found that often online questionnaires take longer for participants to complete than simple pen and paper, as they have to get used to using the online tool – for some reason there appears to be extra cognitive load when completing an online questionnaire.

Tied into the need for speed of completion during sessions is the idea that questionnaires need to be quick to complete to avoid participants half-heartedly ticking boxes because they are bored and want to play the next part of the game, or go home. The questionnaires also need to be efficient, because lab time is expensive, and we want to be as cost effective as possible. Participants do not want to receive a lengthy wad of a questionnaire; they want to move on to the next game, or next section of play. Longer questionnaires can lead to participants rushing through them without paying much consideration to the questions – to counteract this, we can talk the participants through each question as they answer it, and discuss their responses, but this often takes even longer.

Hence, it is worth ensuring that questionnaires are as short as possible. Usually they do not need to be long, because they should be based on the key research questions. There are often only a few specific research areas that benefit from the quantitative data that the questionnaires provide, and these are discussed with the client before we plan the test procedures – typical question areas are listed later in this chapter. As an example, the sessions described in the case study needed adequate time in them to play through 3–4 levels per participant, and so the questionnaire needed to be quick, to fit everything into the 60-minute sessions.

A useful way to reduce a questionnaire's length is to use the questionnaire as the skeleton for your research; the framework upon which you can investigate responses in more detail with other methods. For example, although the questionnaire during the test session may be short, you can pick it up at the end, and investigate the reasons behind participants' answers using a semi-structured interview. An example process could be outlined as follows:

- You keep the questionnaire short
- It ensures you get at the issues that are most pertinent, by eyeballing the extreme ratings as you pick up the questionnaire
- You can delve in to interview the participant about their more extreme ratings when the gameplay part of the session is over

It is essential that questionnaires be administered at appropriate points during testing. Ideally, they need to occur at natural break in the game – at the end of a level, at the end of a minigame, or just after the participant dies (I mean when the participant’s character dies in the game, not the actual participant – dead participants are difficult to get responses from). These sorts of points form natural breaks in gameplay. There are several advantages to probing for feedback at these stages. One of such advantages is that participants are more likely to remember their experiences around that time in play, so questioning them at that time is a great opportunity: any ratings surrounding pertinent research questions will be in the forefront of their mind at that time.

Pen and paper is also a more flexible tool to use during a study than a laptop or a PC. When participants turn to complete the questionnaire, it can break the flow of the game. This is especially the case in family games, or party games, such as *SingStar* (Sony Computer Entertainment Europe, 2004), *LiPS* (Microsoft Game Studios, 2008), *Buzz! The Music Quiz* (Sony Computer Entertainment Europe, 2005), or *You’re in the Movies* (Codemasters, 2008). Often in these types of games, where people are moving about frenetically, sitting at a PC almost requires a change of mindset to ‘work mode’, as they sit and use the refined, small gestures of the keyboard and mouse after the often grander gestures of a Wiimote, Move or Kinect. Alternatively, a paper questionnaire can be picked up, quickly scribbled on, and put down again, without pulling the participant out of the game mindset too drastically. To keep the feedback style with the game style even further, you could also try using response forms drawn on large pieces of paper on the walls, which require participants to mark their responses on them with a marker pen. These big gestures they perform to mark their responses reflect the big gestures they are intended to perform in the game, and keep them in the same frame of mind between plays.

The design of questionnaires and other psychometrics, and how to try to ensure that response biases are reduced as much as possible, is something that is covered in other excellent textbooks and papers, such as Shaughnessy et al. (2008) and Kuniavsky (2003), but which is outside the scope of this chapter. For more resources, please see Oppenheim (1992) and Jackson and Furnham (1999).

One area of questionnaire design that is relevant to this chapter is the understanding that people need a reference point against which to rate something. For example, their responses to open questions such as ‘please can you rate this game’ using a scale of 1–7 will vary wildly between different participants; it is almost a meaningless question in isolation.

When trying to gather information of this kind using questionnaire data we often try to create a ‘grounding’ rating to begin with, to allow participants to give

a score against which they relatively rate features of the game they are currently playing. For example, we can first ask participants ‘what is your favorite game of this genre?’ and then ask them to rate that on a numerical scale. This creates the grounding score – a baseline. Then we can ask them ‘Compared to the game you rated above, how would you rate the current game you are playing?’ and ask them to rate it on the same scale. You then get the optimum score of their favorite game in the genre, and can see how their response to the game they are testing compares to it. This method means that we get a relative score for the game being tested that is grounded against the participant’s existing experiences. It tells us how the game measures up against the participants’ ‘best in class’. We then complement this score with more qualitative methods to understand why it achieved the score it did, and how it can be improved.

23.4 Questionnaire Drawbacks

Questionnaires have some drawbacks too. They are suited to being administered after short levels or short bursts of play. Questionnaires after longer playthroughs can mean that the moments of intense fun or frustration are averaged out in the rating of the player’s overall experience. However, they can help to gather a good overall feel for how players respond to an entire level, mini-game or section.

Responses from the participants are subjective and can be subject to response biases, such as the willingness to please the facilitator. The subjectivity is another reason why we do not just rely on the questionnaires alone, but combine the scores with observational data from usability testing too (i.e. combining multiple measures covering the same phenomenon: here a user test). If we notice a disparity between the body language of the participant and their questionnaire scores (say they look bored and disinterested, but rate the game as excellent), we can chat to them about their scores, and try to understand the reasons behind their ratings. Not to try to change their mind, but to check why they rated and scored the game as they did.

If testing a longer game, like a Role-Playing Game (RPG), where natural breaks are less frequent, questionnaires are more difficult to administer without breaking the participant’s concentration and involvement in the game, and other methods may be better suited. Where natural breaks are infrequent in the game, one has to consider the benefit of breaking into the game play to ask the user to rate their experience. Waiting until later in the session to question a participant may mean that they have trouble recalling experiences from earlier.

Questionnaires can also be too technical for younger children to understand. Reading levels, intelligence, and emotional development all vary greatly between individual young children, so it is difficult to say at which age a questionnaire is not feasible for use. However, we broadly find that children younger than 8 do not have a good grasp of the intricacies of ratings – they will tend to rate at the extremes of any scales. They also have difficulty understanding negatively worded questions (included to try to balance out response bias). Therefore, we tend to avoid this level

of structured response scoring for younger children, even if we, or their parents, are available to help them.

Therefore, questionnaires may not be perfect and suited to all research occasions; research is all about choosing the right tools for each job. Often, we find that using questionnaires in conjunction with other measures, such as observation, semi-structured interviews, discussion, or biofeedback, provide a simple and efficient way of understanding where players are experiencing different levels of entertainment, and allow us to focus in on the ‘why?’. They are also useful tools for benchmarking between and within games.

23.5 Questionnaire Reporting

A client recently asked us to measure how engaging their game was over a period of several months of development. The client had an idea of what they wanted as an outcome (a measurement of difficulty of each of their game’s stages), but they left it to us to decide the best way to implement the research in order to measure and benchmark such a nebulous concept. The research needed to provide them with solid research feedback that they could implement quickly, so that alterations could be fitted into their development timescales as soon as possible. Our feedback needed to be in a format that they and their teams could understand and disseminate quickly and simply.

We work with many different publishers and developers, and nearly every client has their own way of working, and each team often has their own in-house technology for:

- Recording usability and/or QA issues,
- Marking areas of concern during testing
- Grading and prioritizing the severity of issues
- Discussing ways of implementing recommendations uncovered during user research
- Deciding which team members will own particular actions resulting from the research

We would love to have a technological solution that could upload a data file to each team in a format that each could easily access and act upon. In practice, the logistics of creating a system are greater than the value that we would gain, at least in the short term. Instead, a simple spreadsheet of key findings, scores and prioritized recommendations usually meets the client’s requirements, and if they require alternatives, we work with them to meet their needs. An example spreadsheet is shown in Fig. 23.1.

We like to rely on techniques that are as simple as possible as long as we get the best results for the client. By this we mean that the time taken to collect and analyze the data is far outweighed by the value of the data to the research. There exist many different methods for running game user research sessions, and each session needs to be tailored individually to the requirements.

Copyright © 2013 PlayStationGames

Number	Issues?	Recommendations	Severity (H, M, L)	Comments	Sort by priority
1	In all the Levels we've tested, we've found when most people want to move fast, they move the stick quickly. The idea of pressing X is counterintuitive to them. "I found you run faster if you wobble rather than if you keep pressing X fast."	We would recommend detecting waggles of the stick, when gauging speed of running.	H		
2	The X sometimes switches to left handed without meaning to, after celebrating a X.	Although we have heard that introducing a pause between shooting (shooting made the game feel slow), we would recommend looking into this again, as the consequence of going left handed appears to be worse than having a slightly slower game experience, and there may be a way to prevent the switching of body position without losing game pace. Alternatively, assume with the weapon in the same handedness as before any auto-aiming (i.e., remove the option to change handedness during an enemy, it will probably be an uncommon use case anyway).	H		
3	People still didn't understand the moving target when they were aiming. Some people thought they had to line their body up in front of it.	Change the icon to simply an arrow pointing down. The text doesn't have to stay on the screen for long and could disappear more quickly at higher difficulty levels.	M		
4	Team change score change screen requires a 'Skip' button. Players tended to try and skip past it.	Consider implementing a skip button.	L		
5	This time around, people picked up on where to aim using the small text under each player's name.	N/A	Good		

Fig. 23.1 Typical spreadsheet of key findings. High, medium and low priority issues are color coded, and labeled H, M and L, next to the issues and recommendations

Questionnaires form one of the simplest, and most frequently used, methods of gathering quantitative data on user studies. In addition, questionnaires can form a basis of further discussion with the participants, as outlined above.

23.6 Creating a Questionnaire

In order to decide how to create a good series of benchmarking questions, it is important to know the planned development of the game, its key features and sometimes particular marketing angles (e.g. this game has a fantastic new way of controlling your character). This helps you to include the most relevant questions in the questionnaire (e.g., if the marketing is keen to promote the new way of controlling your character, the questionnaire contains specific questions that probe user reactions to it).

It is also important to know what features will be added in the production schedule, and roughly when. This allows you to plan questionnaires, so that you can gather scores early on, to see how they change when new features are added. For example, does the new trick-shot add depth to the game, or is it just confusing and detracting from the core gameplay?

As an example of planning for future game features, if the game is due to have certain features in it in later builds, which are missing from earlier ones, like certain sound effects, or a new type of interface, or maybe a scoring mechanic (such as picking up gold), or different costumes, it might be important to ensure that even the early questionnaires refer to these features, or their intended effects, from the start

of the first round of testing. This allows us to track the effect of these additional features as they are introduced on the players' ratings of the game.

At ExperienceLab, we spend time validating the questions we use in our questionnaires to ensure that they broadly generate the responses and scores that we would anticipate from different game types and game experiences. The process of validation basically runs as follows: Each time we test a new game, we add our data from the questionnaire into our database, and then run a regression analysis to see if the participants' responses correlate (using e.g. factor analysis and correlation analysis) with each of the different questions, implying that the questions are all measuring the same underlying features. Inconsistencies have allowed us to adjust the questions on our questionnaires to focus on those that provide valid and accurate measures of participant responses for particular genres, and to remove those questions that are less predictive. Also, since we often use other measures of behavior response, such as observed participant behavior and attitudes, we run analyses to ensure that all our obtained measures are consistent with each other, implying that they are measuring the same underlying trends. For examples, we could run an analysis to confirm that our observed ratings of the player's body language (such as the amount of effort they are putting in to play the game) correlate with their questionnaire scores of how engaged they are with it.

In general, good areas to consider when designing a questionnaire to evaluate a game are:

- The overall rating of the game (from Terrible to Excellent, or other appropriate descriptive terminology for the audience you are testing – e.g., Awesome, Sick, Gnarly...)
 - A comparative rating first, e.g., “What is your favorite game”, or “What is your favorite role playing game”
 - “How would you rate that game on a scale of 1–7, if 1 is terrible and 7 is amazing?”
 - Then “How does this game compare to the game you just mentioned?”
- Music
- Sound effects
- How much fun it was to play
- Responsiveness of controls
- Perception of time passing
- Understanding of the scoring
- How difficult it was and...
 - How enjoyable it was to play at that level
- Understanding the objectives
- Desire to play again

We tend to use the typical 1–7 Likert-type scales for our ratings, as do many user researchers. A Likert scale is shown below in Fig. 23.2 Participants rate how much they agree with a statement, on a scale of 1–7. One end of the scale is negative towards the statement, and the other is positive.

Q5. I found the game **FUN** to play

1	2	3	4	5	6	7
Strongly Disagree			Neither agree nor disagree			Strongly agree

Fig. 23.2 An example of a question and its 7-point Likert type scale

23.7 Questionnaires for Benchmarking

Questionnaires have provided particular benefit to our research when benchmarking player experiences over time. These can be across different builds of the same game as it develops; different levels within a game, or different games within a genre. I describe some of our experiences in this section; then, I will use a case study to illustrate.

23.7.1 Benchmarking Across Games

As a long-term strategy, at ExperienceLab, we keep a database of the games we work on, and the scores that each game receives on the different questionnaire questions. This is something that many user research organizations do. We do this for several reasons:

1. It enables us to keep a log of the questions that we use in each survey.
2. Of those questions, we can see how certain ones correlate to final review scores of the game, and in a sense can be described as ‘accurate’ predictors. We are constantly trying to validate the measures we take.
3. We can see which questions in a questionnaire produce accurate scores for different genres of games, so that we can vary the inclusion of different types for different genres, and include the questions that are the most relevant and accurate. This helps to ensure the questionnaire contains only the most relevant and useful questions, thereby keeping it as short as possible for the reasons mentioned above.

There is a fundamental assumption with the second and third reasons above: validation can only recommend relevant questions for a new research project if the type of game in the new project is qualitatively similar to the game we have scores for in our database.

Within the same genre not all games are the same. For example *SingStar* (Sony Computer Entertainment Europe, 2004) is not *exactly* the same as *Lips* (Microsoft Game Studios, 2008). There is some overlap: they are both party games, they both require singing, etc., but they are not exactly identical. There are plenty of differences between them, especially across the wider game experience. So this process of matching questionnaire questions to the style of the game is still a rough rule of thumb, but it does help to focus on the key predictors of performance over others.

We do not benchmark questions across genres often, as this would be comparing qualitatively different games, like comparing apples with oranges. However, it could be considered for when certain games in different genres have a similar mechanic.

23.7.2 *Benchmarking Across Different Builds*

We are often asked to work with a title in development, around alpha, and then to track its progress as it is developed. As the game develops, and existing features are refined, and additional features are added, developers can get an overview of how the game is progressing, and whether it is becoming more fun to play.

In the example chart (see Fig. 23.2), we show how the scores for the levels within a Nintendo DS title varied as the game was tested on different dates. It allows us to see how the ratings of what the overall appreciation of the different levels changed as new features were added and the AI was tuned. We are unable to divulge which title it was, because of the non-disclosure agreement we have in place with the client.

The chart indicates that not all levels were tested at each test session. We tested 3–4 levels in each round. However, by keeping your benchmarking methodology the same in each round, you can compare the relative scores of each level at each test point to identify key areas of concern. From this the client can understand where to prioritize their resources.

The research findings comprise these scores, but they also need further information to help us understand the reasons behind the participants' opinions; questionnaires on their own can only reveal so much. However, by triangulating the data from the questionnaires with rich feedback from post play interviewing, and observations of them during play, you can understand *why* players rated the games as they did. You then discover the issues that caused lower ratings, which can then be addressed.

There are often moments in the testing where the importance of not using the questionnaire in isolation becomes apparent. Observation of player behavior and discussion of their experiences is essential. For example, it may be the case that as more features are added, the game becomes more engrossing: there may be a wider range of moves available, or of gameplay features. This could be exactly the sort of feature list that players of earlier builds of the game requested, meaning that the game's ratings are increased. However, at the same time, these new features could be incredibly hard for some players to perform; they may not yet be balanced properly, making the game frustrating. This would have an overwhelmingly negative effect on questionnaire scores that override any benefits that the game might have gained from their inclusion.

This is illustrated in the line for level number six in Fig. 23.3. In the second round of testing the client added new features, including obstacles to avoid, which increased the depth of the game. Additionally, they tweaked the AI and accidentally made it

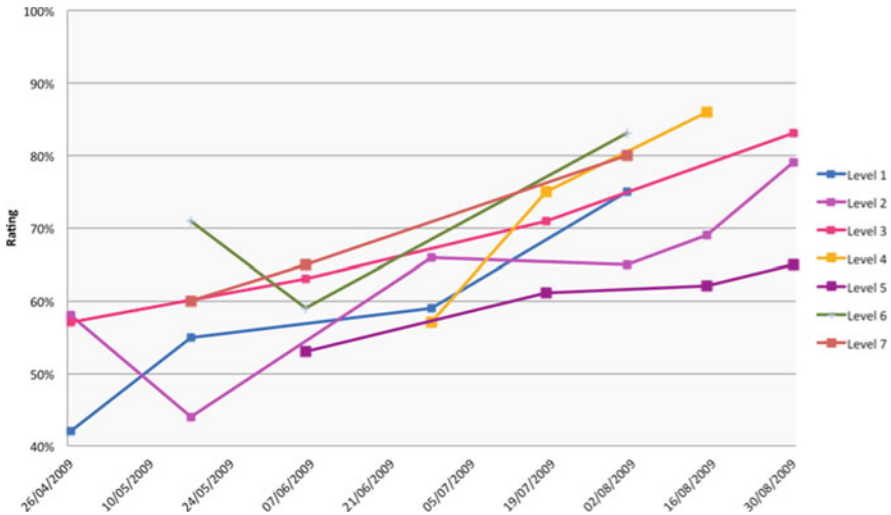


Fig. 23.3 Chart showing relative benchmarked scores of a Nintendo DS game levels across test sessions. The abscissa shows the dates at which each level was tested. The ordinate shows the overall rating that each level received at that time

impossibly hard to beat the CPU. If we were not observing the sessions to see this, watching the player’s body language and expressions as they play through, and coordinating these measures with informal interviews after each session, the questionnaire scores could have led us to assume that the issue was just the new obstacles, and overlooked the AI issue.

But even if in one build the questionnaire scores drop, as described above, the scores are still valuable, because as the title starts to near its end point, the new features are refined further and the game is increasingly balanced, we can provide a broader picture of how the title is improving. Vitality, if a new feature is added late in the day and the game ratings bomb, and there isn’t enough time to refine that feature before submission, the ratings can give the developers a good indicator of whether to keep it in, or remove it (along with their own judgments and other data too).

A key element that relates to this is the user interface (UI) and menu wrapper, which might join up various mini-games in one big party game title. The UI is a more functional element of the game, and is different to test the mini-games, but it can have an effect on the overall appreciation of the final package. In early builds the UI may not be present, and we might be working with just the minigames. But when the UI is implemented, it creates a more polished and professional game package, which affects the player’s appreciation. By checking with the developers about if, when and how this UI might be implemented before the research commences, we can make sure we have a series of questions lined up to handle the overall appreciation of not just the UI, but the complete game product, rather than just focusing on the various mini-games.

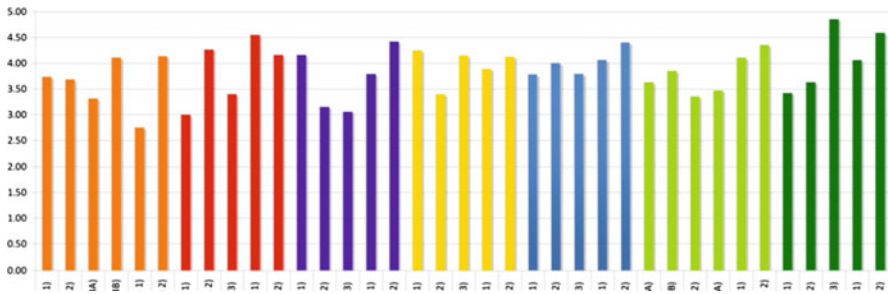


Fig. 23.4 Ratings of level difficulty. Each *colour* area is a ‘world’ in the game, and each *vertical line* is the average rating of difficulty for a single level. The taller the *line*, the greater the rating of difficulty. You can see that each world comprised several levels, and the broad aim was to ensure that each world had a steady increase in difficulty as players progressed through its levels (© Sony Computer Entertainment; courtesy of Sony Computer Entertainment)

23.7.3 Benchmarking Across Different Levels

Benchmarked scores are also a great tool for ordering levels within a game according to different criteria, such as ratings of fun or difficulty. For example, we worked on a multi-level platformer on a handheld device for a major publisher, where following completion of each level the participants completed a short questionnaire, which measured multiple aspects of their attitudes towards it. In particular, the ratings of difficulty, and also how they related to the participants’ enjoyment of playing the level at that difficulty, were used as key benchmarks to compare levels against each other. These were then plotted on a chart to help us understand if the levels were increasing in difficulty as expected. It helped us to understand either where some levels might be swapped in order of presentation, or if they needed to be increased or decreased in difficulty. This helped the game have an appropriate challenge curve as it progressed, and to remain fun. The chart in Fig. 23.4 shows the ratings of difficulty of each level. The game is comprised of ‘worlds’, and each colour shows a world. Within each world there are levels, and each level is a vertical line of the graph, indicating the average rating of difficulty across all participants. As you can see, in the first world, levels 1, 2 and 3 were rated as increasingly easy, but level 4 was much harder. By using the data, in conjunction with viewing participants as they played the level, we could see one particular area in the level where participants got stuck. The developers also saw this, and the level was refined so that the level was less challenging but still fun. Subsequent ratings of difficulty showed that the issue had been addressed. We also took ratings of how enjoyable the levels were. After the level had been altered to make it less challenging its rating of enjoyability increased.

23.7.4 *Benchmarking in Different Countries*

Another useful reason for benchmarking is to understand the relative appreciation of a game across different locales. These can be intercontinental differences, inter-country differences, and even intra-country differences, where a game might appeal in differing amounts to different states or areas.

Consistency is the key across benchmarking generally, but when testing across larger areas it is especially important. There are different methods of obtaining consistency. We find the best procedure is to run the first set of studies in a country that is your own (in my case: England), or as similar to your own as possible (in terms of the language spoken). This allows you to ensure that the questionnaire is picking up the aspects you are interested in from the test sessions. It also allows you to ensure that you have a consistent approach to administering the questionnaires, and we document the required approach in detail to allow other researchers in different locales to follow it precisely. This will then be used as the session guide in other locales.

We primarily get asked to provide services in other European countries, with the occasional foray further afield, to the US, Australia or Japan. We have partner agencies across multiple countries, so the consistency of approach and the expected outputs are understood up front by each partner, which minimizes the briefing time.

We recommend using partners who are based in the country to run the sessions. It is better to have a local facilitator in each country than it is to have the same facilitator running sessions in multiple countries. The local facilitators in non-English speaking countries can help you to word questionnaire questions to convey the nuances you need. They are also able to interpret the answers to more qualitative questions within the framework of the research objectives.

There are various means of conveying the testing requirements and procedures to the partners to ensure consistency. They vary in cost and affect the turnaround time of the research; so again, it is always worth discussing the pros and cons with the clients, to help them make an informed decision. Here are the typically different ways that you can increase consistency, in decreasing order of vigilance:

1. Travel to the locale and brief the partner agency in person before the sessions begin. Clearly convey the requirements and procedures and stay to observe all sessions and take notes.
2. Brief the partner agency in person, watch the first one or two sessions to ensure consistency, take notes and then head back as the remainder of the sessions take place, run by local researchers.
3. Brief the partner agency by phone, Skype or email before sessions commence, and ensure that the sessions are running smoothly by observing a session over IP.
4. Brief the partner agency by phone, Skype or email, and then call the partners after the first session to verify how it went. Also eyeball the data received back from the questionnaires, to ensure it matches expectations. If not, discuss any

differences with the partner, to understand how they arose, and particularly whether they are an accurate reflection of attitudes or an undesired artifact of the test procedure.

We regularly run these international sessions, and differences between cultures and locations are always interesting to pick up. Questionnaires provide the initial triggers to make us aware of potential differences, but then we still need to back them up with further discussions and interviews with participants, to understand the reasons *why* there are differences. Because of this, it can be important to have a steady stream of data being returned, or monitored and compared to locales who have already completed. That way any differences can be probed further in the remaining sessions.

23.7.5 Summary

Questionnaires work well as a method for understanding how titles compare against other similar titles, against versions of themselves as they develop, and across different locales. They can help you pragmatically provide solid feedback to your clients. However, they are valuable beyond just the pragmatics: the data they provide is useful in both the short and long term for benchmarking.

They are simple, inexpensive, portable and transferable, and they provide a wealth of data that can trigger further investigation using other methods. Their outputs are flexible, they do not require upgrades to maintain working, and you can transport them across continents easily. You might not want to use them in isolation, but instead use them to complement and backup findings from other methods. They often feel like the framework of the research findings, with the more detailed qualitative findings allowing us to probe deeper.

There now follows a case study of some testing we recently performed for a UK games developer. The study was designed to monitor the players' ratings of 'fun' across multiple builds of the game. We hope that the case study illustrates how we work as a consultancy with clients, understanding their requirements and creating a research strategy. It should also illustrate the overall process of running user testing sessions, and practically how questionnaires and surveys fit into the battery of methods available to the researcher.

23.8 Case Study

23.8.1 Client Requirements – Our Brief

Due to the confidentiality issues surrounding our work, and the non-disclosure agreements in place, we are unable to talk about which company is involved in this case

study. However, in this section the procedures that we employed with the UK developer and other companies are the focus of the case study.

At the beginning of each project, the client's requirements from the research are usually conveyed to us in a brief. This is usually a formal written document, which we then discuss with them verbally. The goal is to ensure we provide them with answers to their research questions at the end of the research.

The client wanted ratings of participant enjoyment of the game. They wanted them for each build of the game we were given, so that they could get an idea of whether they were going in the right direction as the title developed. They wanted a guide to know if the new features that they introduced at each test point were increasing the participants' enjoyment of the game, or if they needed further refinement.

In brief, the game was based on the Nintendo DS, comprising several levels and aimed at 9–15 year olds. It had an aspect of social trading to it. The client had a well-developed idea of their audience for the game, and wanted to understand how each level appealed to people generally. They also wanted to understand the appeal of the game to hardcore gamers vs. more casual gamers, and across some age band distinctions. The client asked for measures of user attitudes to the levels they were developing, broadly along a scale of positive to negative. We used a range of measures, across different dimensions (outlined below). The key requirement was that at each round of testing, we took consistent measures, so that we could compare the game levels relative to previous ratings.

In terms of process, the client wanted to provide us with a series of builds every month, and gather feedback on each level for each build. This enabled them to have set deadlines for each build to be ready, and for us to plan our resources in terms of availability, test labs and participants. Time was a massive pressure on the client, so they needed the major issues reported back to them as soon as we picked them up, and the final scores back to them as soon as possible after the sessions completed.

We initially asked them if they could consider using the RITE method (Medlock et al. 2002). The RITE method is where the major issues that are quick fixes are addressed as quickly as possible by the developers, often after one or two participants have taken part. The developers then produce another build for testing as soon as possible during the sessions (sometimes in time for the next session). This leads to a swift evolution of the game, with research findings being acted upon as soon as possible. However, the client could not incorporate this method for two issues:

1. They needed to keep their developer resources as focused as possible on the roles they were currently working on. They were not able to assign resources to stand by for ad hoc fixes and alterations that the RITE technique would require.
2. We were testing away from their site, and we would need to transfer large amounts of code between our two locations in order to transfer a new build to us. There was not enough time between sessions to transfer the code of a new build.

All in all we needed to run the sessions and get the client the ratings of enjoyment as quickly as possible, plus video, plus qualitative feedback about how it could be improved.

23.8.2 How We Approached the Brief

In a sense this was a relatively simple brief. The client knew what they wanted to test, although they did not know exactly which levels would be available at each test point. They also knew when they wanted to test, as they could provide builds of the game to us every month, with the dates set well in advance. They had time to create eight iterated builds for testing, 1 month apart, before they hit an internal development deadline.

Following some discussion with the client about how long each session should be, based on the amount of content in each level and estimated playthrough time, we decided that 45 min of the test sessions would be adequate. This would give each individual participant around 15 min with each level.

Given these constraints, we provided a series of quotes for the research. The client understood that broadly speaking, the more participants we get for these studies, the more reliable the data from the questionnaires would be. Thus, we created quotes that varied the numbers and types of participants, allowing them to choose the optimum tradeoff between participant numbers and costs. As we mentioned at the start of this chapter, the client knows best, and they know their budgets.

Meanwhile, we knew that we needed to start focusing on creating the materials to record the participant feedback. These were the key features we focused on, which are described in the next subsections.

23.8.2.1 Our Primary Function Was to Provide an Overall Rating Score of Each Level

However, we did not want the ratings to be the only feedback we provided. We would be observing sessions anyway, as would the client, and there would be a wealth of information we could also include to support and understand the feedback scores better; to make them more meaningful and to provide ideas of how to improve them in future iterations. So whilst we would concentrate on the quantitative rating scores, we would not ignore the qualitative, and would aim to use it as part of the feedback as well. As we mentioned earlier in this chapter, the questionnaires formed the skeleton, or structure, and the qualitative feedback allowed us to understand the ‘why?’ of the answers and actions.

23.8.2.2 Questionnaires, Interview and Observations Seemed Appropriate

As mentioned earlier, questionnaires work well after short bursts of activity, and after each level there would be a natural break point, suitable for administering questions to participants. We did not want to rely solely on self-reporting, so we would also complete a form of structured observations for each player, rating their responses on a series of scales while observing from behind one-way glass.



Fig. 23.5 Participants taking part during testing

We would also look for more qualitative issues, providing feedback on observed issues of gameplay that were causing frustration (or enjoyment). Some sessions had children as young as nine in them, so we needed measures that were simple and easy for youngsters to understand. Finally, after the participants completed their questionnaires we spoke to them about their responses, to understand the reasons behind their answers.

23.8.2.3 The Client Needed Feedback As Soon As Possible

We knew that the client needed the feedback quickly, and in a form that could be easily understood and distributed among staff as soon as possible. We had no time for processing of data, other than entering ratings into a spreadsheet and performing simple calculations. Thus, simple charts were exactly what was required. We also encouraged the client to watch sessions in our studios behind one-way glass, so they would be able to feedback issues to their team as and when they occurred, by email, phone or instant messenger. An image of participants taking part is shown in Fig. 23.5.

Our feedback was designed to present the quantitative data, but to also suggest recommendations from an independent, objective perspective. For these reasons, we created a feedback spreadsheet of scores, observed issues and recommendations for each level, which could be emailed to the client the day after the last session, simply and quickly.

We had a battery of the following measures:

1. Participant questionnaires
2. Observer ratings of the children playing, looking for body language and gestures to indicate positive or negative emotions
3. Qualitative observations
4. Participant interviews

We used these different measures to try and eliminate as much response bias as possible. The observer ratings would allow us to factor in our own perception of the participant's enjoyment of the game, from observation of body language, and reactions to gameplay. Our qualitative observations were based on observations of moments of frustration. Participants' interviews also enabled us to learn more about their experience and the reasons for their questionnaire ratings. These different measures enabled us to record the reactions of users, which we then combined to create a final score, calculated with the data from all the participants. We could then provide the client with the information they needed, as soon as possible, in a format that was easy to distribute.

23.8.3 Creating the Questionnaire

23.8.3.1 Participant Questionnaire

We needed a questionnaire for participants to complete after they had played each level. Using our database of previous questions, mentioned above, we could isolate questions that had previously been good predictors of social/trading game scores. From this we pulled together a list of the key questions to include. These included:

- An overall rating of the game
- Ratings of the graphics, music, sound effects
- Questions of control response
- Perceptions of how much time had passed
- Understanding of scoring
- Ratings of difficulty, and how enjoyable that difficulty was
- Understanding of objectives

We knew from our previous research that these question areas contribute well to review scores. After each piece of work is complete, we take the data from the ratings scores and run it through a regression analysis to see if all the key questions are contributing in the same direction to the final rating score that we produce. We also added some other questions to gather information related to specific areas of the brief, such as 'how easy is it to trade items?' After this research was complete, we did the same regression analysis on the questions in this questionnaire, and found that there was a high degree of correlation between the scores on each question,

indicating that the questions were measuring the same general parameter, and contributing collectively to the final score.

When creating the questionnaire we also had to consider that there are younger children completing it too. Thus, we needed to make it short and simple. Because we had an idea of what the most pertinent predictor questions to ask were, we were able to create a concise questionnaire. It was short, and all participants could complete it quickly between levels.

23.8.3.2 Observer Ratings

We wanted a consistent measure of observed participant behavior across the rounds of testing. We could then combine this with the self-reported questionnaire data to produce a more reliable set of scores. Triangulating our ratings like this, using multiple measures of observation as well as self reporting should lead to more reliable data. Often participants tend to rate gameplay more favorably than we would expect them to, given what we observe of them as they play, and children are especially prone to rate things highly. So our observer ratings combined with the players' ratings to provided a more stable score.

The ratings sheet had several elements to it, for the observer to rate on a continuous scale. They included:

- The level of engagement the player exhibited.
- Whether their progress through the game appeared to match their expectations and requirements.
- The reactions and utterances to the game during play, whether positive or negative.

We also investigated participant reactions to being interrupted during playing. The idea was that after a set period of time, the facilitator would enter the room during a game and ask them to stop playing. If participants were enjoying themselves, they would be disappointed to be interrupted, and if they were not having a good time, they would be relieved. We would then mark their reaction on a scale, from positive to negative.

23.8.3.3 Further Ratings

We also wanted to provide a simple feedback format for younger participants, but we extended its use for all participants too, for consistency across them. All participants completed all questions, and were rated on all aspects.

We used a simple measure shown in Fig. 23.6. This was a measure that asked participants which of three they felt most represented their mood after playing the level. One was a smiley face, one was sad, and one was in-between, which the client called "meh". The smileys were simple, and easy for the users to rate, especially the younger ones. We frequently use these smiley ratings when working with children, and adults find them a fun way of responding too. It's a broad measure of their attitude, but they form just one strand of multiple ratings that we take.

Fig. 23.6 Typical face images that we use to help young children express their opinion of a game



These were the main quantitative measures we took. We also supplemented them with the more qualitative usability observations and recommendations. Our interviews with each participant after all the levels also allowed us to dig deeper into particular reactions and responses.

The scores from each question were recorded in a datasheet, normalized, and combined in a particular way, which we refined during the pilot phase. This combination of scores allowed us to create a score for each level that was sensitive enough to alter when small changes were made to the builds. However, the score was also steady enough to ensure that it did not hit a ceiling point, where any further improvements to the game were unable to have an effect on scores.

23.8.4 Items We Refined After the Pilot Studies

We had factored in a pilot round of testing to ensure that

1. The ratings were calculated in a way that did not lead to ceiling effects.
2. The ratings were sufficiently sensitive to pick up smaller changes in participant attitudes.
3. Participants understood how to complete them.
4. They were simple enough for us as researchers to keep track of and administer.
5. They didn't take too long, and we didn't run out of time.

The pilots also allowed us to refine the calculations and combinations of scores. There were a few things that we altered, as follows:

- We changed the interruption score so that it always occurred between levels, not during a level. When we interrupted during a level, the response to the interruption was always biased by how far through the level the participants were. For example, if they had only just started a level, they usually did not mind an interruption. To keep the measure consistent between subjects, we interrupted them between levels.
- From the observer's ratings sheet, we removed the measure of progress; of whether the participant was making suitable progress through the level. This was because it was hard to get a consistent rating of this between observers, and the idea of what constituted appropriate progress was hard to define.
- We included a new question in the questionnaire to ask people how likely they would be to play the game again.

23.8.5 Case Study Summary

These scores allowed us to create a quick synopsis of progress through the builds, for each level, and were supplemented with a rich picture of participant reactions from the qualitative data. The client was pleased with the data that they received from the research, and mentioned that it also added a sense of competition between development teams working on different levels. The data that came back was useful to the client, not just in terms of it adding to a sense of friendly competition between developer teams, but because they could see instantly if a level was not up to the standard of others, and focus resources to address it.

After five evenings of sessions, totaling 20 participants per round, we sent to the client a spreadsheet containing:

1. Quantitative raw data for each of the levels.
2. The scores aggregated and calculated to provide a final percentage score for each level.
3. The overall percentage score compared across each build that we tested.
4. Charts to illustrate the differences on each level between
 - Ages of the children
 - Casual and hardcore gamers
 - Those that defined themselves as ‘social gamers’ and those that were simply ‘gamers’
5. Qualitative findings and recommendations for each level, prioritized, to help improve the score in future builds.

The final ratings and feedback were sent to the client very quickly after testing completed, and the data had been entered into the spreadsheet. This was usually sent to them the afternoon after sessions completed, allowing the client to see the feedback quickly and disseminate around the team easily.

Across the builds that we tested, the scores gradually improved across all the levels, allowing the team the confidence to feel that they were moving in the right direction as they implemented new features. We have developed this methodology since then, adding in new features where the game genre requires it.

23.9 Conclusions

I wrote this chapter to highlight that reliable and sensitive measures of user reactions to games can be obtained through inexpensive and simple sets of questionnaires and ratings – something that we use frequently at ExperienceLab. These measures can be used to create a series of benchmarks that allow developers to get an understanding of the standard of their product during development, and how it is developing. Furthermore,

these simple techniques are quick to administer and require little post session analysis.

If anything, the most effort required to run such benchmarking sessions is up front, before the testing begins: deciding which measures to use according to the audience age, type of game, time scales for testing, and budget. Then from there, the need to decide which questions are most appropriate to include in the questionnaire, based on your knowledge of how the game is intended to develop and its key features. A key takeaway point that we would recommend is to just try different elements and measures through piloting.

Pilot sessions are essential, not just if you are trying out new measures to see if the participants understand them or if the returned scores are meaningful and correlate with other similar measures in an expected way. The pilot session allows you to see if the overall scores coming back in are combined in a way that gives the correct weighting to the measures that are most important. It allows you to tweak the approach and gives you confidence that the remaining sessions will run smoothly and provide meaningful data. If you do not have an existing database of questions that you know will be appropriate to test with, and which will provide accurate answers, the pilot sessions allow you to try out new questions, or to observe where the questions are misunderstood by participants. Even though we have performed similar benchmarking studies before, and have a good idea of which questions will work best in a questionnaire for a particular genre, we always run a pilot, because no two games are the same.

One of the drawbacks of not being part of a development company is that despite being formally signed up to a non-disclosure agreement, we are often unable to access other data from the developer or publisher that might be useful for us. For example, at the end of a series of user tests recently, we were looking to validate our benchmark scores, to help us feel confident that, despite the benchmark scores providing a series of reliable relative measures against each other over the rounds of testing, the final scores we produced in some way did show which of the levels would be most popular when the game was released. That is, we wanted to know if the level that scored highest in our benchmark was actually the level that was played the most when it was released. Many game developers and publishers have started to collect player behaviour telemetry from games, as discussed in many case studies and chapters of this book, to allow developers to remotely monitor game play when the game is released and in the wild. It can, for example, allow the developers to understand when to launch a new piece of downloadable content, as they see a critical mass of players gradually reaching a certain stage of the game. However, this data is often too sensitive for third parties to access.

We are still working on a way to access this sort of data ourselves though, with a view to validating the questionnaires scores using freely available online data. Bruce Phillips of Microsoft Studios Research group gave a talk at the Game Developers Conference in 2010 that outlined a potential way that game data could be ‘scraped’ from the Xbox Live website. The talk he gave is listed in the Further Reading section at the end of this chapter. We are attempting to use Bruce’s techniques to understand when players are playing, and what achievements they have attained, and will then

analyze this data with the final ratings that we obtained from testing. This should hopefully give us greater validation that we are indeed providing the accurate and reliable measure that the client expects.

23.10 Next Steps

We hope that this chapter has given you an insight into the use of questionnaires and other measures in the creation of benchmarks, as well as some of the potential pitfalls that could occur. The Further Reading section below lists papers that discuss the construction of robust questionnaires, and which have been a valuable source of information to us.

Further Reading

- Jackson, C., & Furnham, A. (1999). *Designing and analysing questionnaires and surveys: A manual for health professionals and administrators*. London: Whurr Publishers.
- Kuniavsky, M. (2003). *Observing the user experience: A practitioner's guide to user research*. San Francisco: Morgan Kaufmann.
- Medlock, M. C., Wixon, D., Terrano, M., Romero, R., & Fulton, B. (2002, July). Using the RITE Method to improve products: A definition and a case study. In *Proceedings of the Usability Professionals Association*, Orlando, FL.
- Oppenheim, A. N. (1992). *Questionnaire design, interviewing and attitude measurement* (2nd ed.). London: Continuum International Publishing.
- Phillips, B. (2010). *Peering into the black box of player behavior: The player experience panel at Microsoft Game Studios, GDC 2010*. <http://www.gdcvault.com/play/1012646/Peering-into-the-Black-Box>
- Shaughnessy, J. J., Zechmeister, E. B., & Zechmeister, J. S. (2008). *Research methods in psychology* (8th ed.). New York: McGraw-Hill Higher Education.

About the Author

Ben Weedon is the principal consultant and manager at ExperienceLab, previously PlayableGames, a games user research consultancy based in London, UK. He has been active as a user researcher since 2001, and before that was an experimental psychologist investigating language and speech. He has two kids, plays the drums, and is part of the steering committee of the Games User Research SIG (part of the IGDA). He tries to play games as much as his kids, drums and work allow.

Chapter 24

Interview with Simon Møller from Kiloo

Alessandro Canossa

Kiloo is a game and apps development studio based in Denmark, focusing exclusively on mobile phones and has been doing so since 2000 when phones displays had double digit resolutions and two colors, and games could fit in 30 Kb. The company was started by Jacob Møller. In 2008, with the explosion of smartphones, it skyrocketed from 14 to 45 employees. Kiloo is both a publisher and independent developer pushing a new model for co-productions. The breakthrough was in 2011 with the release of *Frisbee Forever* (Kiloo, 2011) for iPhone, the game topped the charts in nearly 60 countries. In May 2012 the company launched *Subway Surfers* (Kiloo, 2012), the title has been in the top 25 iPhone games in the US market for months.

Simon Møller is the CTO, responsible for overall product quality and all the game productions. In the early to mid 1990s, Simon was active in the Scandinavian demo scene, developing his first game when he was 15. He then worked in advertisement for the next 10 years. Simon joined Kiloo in 2008.

Q: Could you describe Kiloo co-production model and how game telemetry is a central part for it?

Simon: In the traditional boxed-product model, publishers bring funding, distribution power, marketing knowledge and quality assurance. In a scenario where digital distribution is the norm, the only factors that still make a difference are market knowledge and funding. Several studios have closed and many more have been resized. There is a multitude of skilled developers that, out of a job, decide to create their own companies and produce their own, often free-to-play, browser or mobile-based games. These startups, although extremely skilled in terms of design and technology, lack the marketing and business intelligence of seasoned publisher.

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

Freemium games are basically open markets: players are welcome to enter free of charge any time and can stay as long as they want. They also have freedom of action, even though some actions have a cost. These actions or items are for sale and the way the merchandise is presented has massive repercussions on the revenue for the game. Killoo looks for strategic co-productions with developers of this kind and offers part of the funding, market knowledge and business intelligence. The business intelligence I am talking about is a game telemetry suite already in place and already fine-tuned in order to be able to capture game data to guarantee optimal monetization. So we don't just offer a solid technological setup but also market knowledge through the variables we have selected, the measures we have defined and the features used in our analyses. We also provide UI design that has been tested to help certain type of games. We provide a set of templates both for metrics setup and interface that have evolved during the many years we have been in the business, and we constantly keep learning and iterating.

Q: What metrics have you defined in your system?

Simon: We have a standard set of measures that we deem important and tend to include as basics in all our games plus a variable set of measures that are define game-by-game, tailoring the telemetry system. The standard set looks at most (and least) popular actions and sequence of actions, most (and least) popular items, beside the traditional daily active users (DAU), monthly active users (MAU), average revenue per user (ARPU), average revenue per paying user (ARPPU), conversion rate (CR) and life-time value (LTV). During development we also offer qualitative user evaluations to fine tune the play experience. Even our standard set of metrics changes according to the particular game in question: some games are geared towards “whales” (few customers that pay a lot of money for a long period of time) while other games tend to attract many more user, paying much smaller amounts for a shorter time. We adjust the metrics collected according to the potential of the game and according to whether we want to “funnel” users or not.

Q: Will Killoo only focus on co-productions in the future?

Simon: By all means no, we are a gameplay company first, so we evolve and iterate the methods I just described with our own productions as well. We take great pride in the games we develop ourselves and it is thanks to those successes that we are able to attract co-productions. Our portfolio is our best marketing tool, if we did not produce successes ourselves we would not be very believable.

Q: In terms of analysis, visualization and reporting, what is your toolset?

Simon: We use intensively Excel and Flurry,¹ which is the industry standard for mobile applications, once the Flurry API is included in the games, all metric data is sent to their servers. We do not perform particularly advanced analyses, Excel can take care of all our needs. The reports are generated by the business intelligence

¹<http://www.flurry.com>

team and shared with different stakeholders: design, marketing, technology, user interface and even artists. On a day to day basis, all the teams have direct access to the data from Flurry, only periodically we generate high-level reports.

Q: What suggestion would you give to mobile phone game developers that are beginning now to include telemetry in their products?

S: Limit what you track and measure; we initially measured too many events and most of them were useless, generating more confusion than real knowledge. It is part of our added value to bring this experience to our co-productions: a streamlined set of metrics to correctly map user behavior in a relevant and timely manner. The worst mistake would be to change or add metrics some time after the game has been released. No matter what, metrics are integral part of mobile game development; it is no longer an option to leave this practice out. Just be weary of not pushing metrics for exasperated monetization, but try to use the tool to improve the quality and the player experience, high quality games have a way to make money.

Q: What's next for Kiloo in the immediate future?

Simon: I strongly believe in our co-production model, we are about to sign up three new teams, one in the US, one in Canada and one in Denmark. I am putting a lot of efforts in the evangelization of our model and I believe it's about to take off. And all the teams we work with, fully realize the importance of metrics for the success of their games, fortunately we do not need to convince them. They might lack the experience to implement telemetry systems with success, and that's what we bring to the table: examining the games, define a set of metrics and provide them with reports and recommendations, besides having access directly to the data.

Chapter 25

Visual Attention and Gaze Behavior in Games: An Object-Based Approach

Veronica Sundstedt, Matthias Bernhard, Efstathios Stavarakis,
Erik Reinhard, and Michael Wimmer

Take Away Points

1. Although eye-tracking can tell us *where* a user is looking, understanding *what* a user is looking at can be more insightful in game design
2. Different levels of abstraction can be used to represent a stimulus in gaze analysis, ranging from pixels, shapes and polygons to objects and even semantics
3. By gaining access to the internal representation of scenes, it is possible to map gaze positions to objects and object semantics

25.1 Introduction

In the design of interactive applications, notably games, a recent trend is to understand player behavior by investigating telemetry logs as is the focus of many chapters in this book or by integrating the use of psychophysics as is the subject of Chaps. 26 and 27.

V. Sundstedt (✉)

School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden
e-mail: vsu@bth.se; veronica.sundstedt@bth.se; sundstedt@gmail.com

M. Bernhard

Institute for Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria

E. Stavarakis

Department of Computer Science, University of Cyprus, Nicosia, Cyprus

E. Reinhard

Department for Computer Graphics, Max Planck Institute for Informatics,
Saarbrücken, Germany

M. Wimmer

Institute for Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria

In addition to these valuable methods, measuring where players are likely to focus could be a very useful tool in the arsenal of game designers. This knowledge can be utilized to help game designers decide how and where to allocate computing resources, such as rendering and various kinds of simulations of physical properties. This leaves as many computing cycles as possible free to carry out other tasks. Therefore, the perceived realism of a game can be increased by perceptually optimizing calculations that are computationally intensive, including physically based lighting, animations (e.g. ray-tracing Cater et al. 2003, crowds of characters McDonnell et al. 2009), physically correct simulations of the interaction of materials (e.g. collision detection (O’Sullivan 2005), natural behavior of clothes or fluids etc.). Level-of Detail-variants of simulation or rendering techniques can be used in regions which are less attended by the player, while accurate simulations can be used within the expected focus of a user. Verifying or improving game mechanics and AI could be other uses.

The study of gaze behavior can provide insight about the visual attention of players and thus assist game designers in identifying problems with gameplay due to a misguided visual perception of the game environment. Moreover, knowing what a player does or does not notice can be used to control the difficulty of a game. For example, the designer may choose to make important task-relevant objects less apparent in the user’s attentional field to increase the difficulty of the game, or accentuate them to decrease the difficulty. Other potentially useful computer graphics applications proposed so far include focus prediction for tone-mapping of high-dynamic range images (Rahardja et al. 2009), the selection of the optimal focal plane for depth-of-field effects (Hillaire et al. 2008) and the minimization of vergence-accommodation conflicts in stereo 3D to reduce visual fatigue (Lang et al. 2010). Further applications include the natural animation of eye-movements in agents (Itti et al. 2006) and estimating or increasing the visibility of in-game product placements (Chaney et al. 2004; Bernhard et al. 2011).

Eye-tracking can be used as a tool to study eye movements or gaze behavior (Duchowski 2003). There are many application areas for the use of eye tracking and it has previously seen extensive use in psychology, neuroscience, human factors, and human computer interaction. Eye tracking devices, commonly referred to as eye trackers, were intrusive and cumbersome to use at the beginning, but recent advancements in eye tracking technology have made it possible to use them effortlessly without distracting users. Although low-cost solutions have emerged, more robust and accurate eye tracking systems are still very expensive. Nevertheless, even with today’s technology the eye tracking process still suffers from various limitations which have an impact on accuracy (Hansen and Ji 2010). Some of these issues are related to the calibration process, the ability to eye track different users, the fact that the eye is never completely still, and the extraction and interpretation of eye movements.

When an eye tracker is used to study gaze behavior in a computer game, the output data is essentially a sequence of gaze points defined by a 2D position on the display screen and a timestamp. With this information, one can establish *where* gaze was deployed in screen-space over time. Analysing gaze data for static stimuli can be time consuming, but it is even more difficult for dynamic stimuli (e.g. virtual environments such as games) (Ramlohl et al. 2004; Stellmach et al. 2010b). A useful representation of gaze data are gaze point density distributions, which can quantify the amount of attention deployed to each region in the display. When a



Fig. 25.1 Example heatmap from one participant viewing game stimuli for 15 s. The visual representation of clustered areas indicate locations of a higher number of fixations

stimulus rendered to the display is static or its changes are very limited (e.g. in web pages), it is sufficient to compute gaze point density distributions in screen space. A prominent tool to illustrate screen-space gaze density distributions is a fixation map (Wooding and David 2002) or heatmap. A heatmap visualizes gaze point densities with colors such that warm colors encode high densities and cold colors low densities. An example of a heatmap can be seen in Fig. 25.1.

For computer games, we have to assume a dynamic stimulus, where temporal changes have a significant impact on the spatial distribution of gaze points on the screen. In this case one cannot accumulate gaze densities in screen space over long time periods because the viewpoint and the objects in the scene may considerably change their positions from one frame to the next. When spatial properties such as the viewpoint or object positions are changing frequently, it may not necessarily be appropriate to analyze *where* a user is looking. Instead, if we consider that semantic properties of scene objects are changing far less often, a more useful approach is to study *what* a user is looking at, especially since the meaning of game objects is supposed to have a major impact on the attention of a user.

25.1.1 *Measuring “Where” But Analysing “What” Users Are Looking At*

To analyze in a dynamic scene what a user is looking at, we need to record the changes in the display during the eye-tracking study. In the subsequent analysis, the recorded data is then used to reconstruct the frames depicted on the display in

temporal alignment with the corresponding gaze data. Thus, gaze analysis tools provide screen recording functions which capture the images rendered to the display during the experiment. The screen recording can then be played back as a video during the analysis stage. A synchronous visualization of the recorded gaze data superimposed on the playback of the corresponding stimuli provides an intuitive clue about the behavior of a particular participant. But with this functionality alone, one can just study the behavior of a particular subject in particular situations. In dynamically changing games, it is unlikely that different subjects are presented the same stimuli while playing a game. To complicate things further, even if the participant partakes in a number of gaming sessions, it is unlikely that the same sequence of in-game events will be triggered to generate the same stimuli. Moreover, encoding *what* a participant might actually be looking at mainly depends on the person who performs the analysis. For many purposes, an objective statistical evaluation, such as computing the gaze density distribution over different objects, might be preferable.

Commercial gaze analysis tools can accumulate gaze-points for manually defined regions of interest. To outline objects of interest in videos, the experimenter has to define regions of interest (e.g. bounding rectangles or polygons) around the objects on a frame-by-frame basis. This can be a tedious and time consuming procedure. To some extent, tools from computer vision, such as segmentation algorithms, could assist in this process. However, translating pixel regions to semantically encoded scenes remains a difficult problem in computer vision.

Fortunately, obtaining a semantic representation of the stimulus is significantly easier for computer games, where information about any game entity can be extracted from the game application directly. Game engines usually render each image from an object-based representation of a scene, from which semantic information can be obtained to a considerable extent. Recording the game engine's internal representations of game states allows the conservation of object-space information of the stimuli, which is otherwise very difficult to extract when only rendered images are available. Therefore, we could use these facilities to map gaze points back to the 3D objects that were observed during a gaming session. We assume that objects are modeled as semantically meaningful clusters of polygons, thereby allowing this approach to link gaze data to object semantics. This chapter outlines such an approach in more detail.

25.1.2 Overview

The remainder of the chapter is organized as follows. Section 25.2 introduces the reader to relevant concepts of human visual attention and eye movements. Eye tracking methodology is discussed in Sect. 25.3. The section also briefly describes some of the related work in studying visual attention and gaze behavior in computer games. Section 25.4 presents some related work in creating 3D gaze data visualizations and logging game data. Section 25.5 describes a unique pipeline that can be used to study

gaze behavior in games, while Sect. 25.6 details the underlying algorithms for mapping fixations to objects. Section 25.7 then discusses ways to use such mappings to collect statistics and draw conclusions. Representative examples of using this pipeline are presented in Sect. 25.8 and some limitations of the work in Sect. 25.9. Finally, Sect. 25.10 summarizes and discusses the work and highlights some of the key issues and important areas of further research in this emerging area.

25.2 Visual Attention and Eye Movements

Humans have different sensory systems which convert information from the environment into neural signals that are then interpreted by the brain. To derive meaning, the brain implements processes that select, organize, and interpret the information from our senses. To enable living in complex environments, humans rely strongly on vision, which consists of two broad components. The first is *perception*, which is pre-attentive. The second is *cognition*, which involves high-level processes, such as thought, reasoning, and memory (Palmer 1999). The delineation between these two is not sharp, and significant feedback and cross-talk exists between the two. When carrying out a task, the human visual perception aggregates low-level features into higher level representations, thus informing cognitive processes while affecting gaze direction. In turn, cognitive processes can guide perception, for instance by actively focussing attention on a particular part of a scene (Yarbus 1967).

Since the information-processing capacity of our brain is limited, incoming information has to be filtered so that we are able to process the most important sensory inputs. Visual attention is the control mechanism which selects meaningful inputs and suppresses those of low importance. Our eyes can sense image details only in a 2° foveal region, due to a rapid falloff of spatial acuity towards the periphery of the fovea. To reposition the image onto this area, the human visual system uses different types of eye movements. Saccades are fast and ballistic eye movements used to reposition the fovea. These movements are both voluntary and reflexive and last between 10 and 100 ms. There is virtually no visual information cognitively processed during a saccade (Duchowski 2003). Between eye movements, fixations occur, which often last for about 200–300 ms (Snowden et al. 2006). During a fixation, the image is held approximately still on the retina; the eyes are never completely still, but they always jitter using small movements called tremors or drifts (Snowden et al. 2006). According to Jacob and Karn (2003), a scan path is a spatial arrangement of a fixation sequence. A common way to visualize a *scan path*, or *gaze plot*, is to overlay a snapshot of the stimuli with fixations drawn as circles, as shown in Fig. 25.2. These circles are interconnected with lines that represent the saccadic eye movements. Their radius can be adjusted to indicate the duration an observer has been looking at that particular point.

The cognitive psychology and neuroscience literature contains a vast array of reports on models that try to predict the mechanisms of attention (Wolfe 2000). The most established is a model which divides attention into bottom-up and top-down

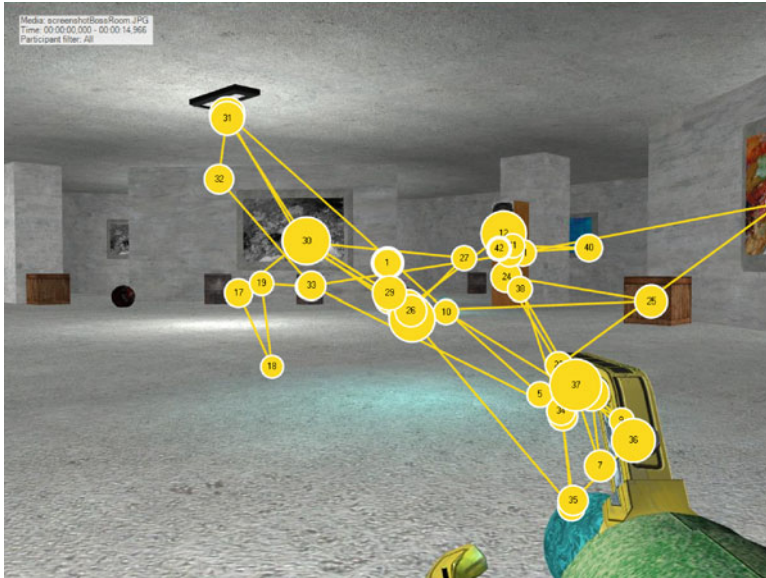


Fig. 25.2 Scan Paths: example scan path from an experiment (Bernhard et al. 2011) with a 3D First Person Shooter game featuring a dynamic camera. The *numbered circles* indicate successive fixations connected by *lines*, which denote *saccades*. The radius of each circle in the latter representation is proportional to the relative duration of each fixation

processes (James et al. 1890). In bottom-up processing, the visual stimulus captures attention automatically without volitional control (Itti et al. 1998). Low-level, bottom-up features which influence visual attention include contrast, size, shape, color, brightness, orientation, edges, and motion. In contrast, top-down processes focus on the observer's goal; they depend on the task. Low-level features in the environment that trigger pre-attentive focus are called *salient*. Features that attract attention as a result of performing a specific task are called *task relevant*. However, bottom-up and top-down processes cannot be separated perfectly, and there is much interaction between both (van Zoest and Donk 2004; Wolfe 2007).

Computational models have been developed which aim to predict which aspects of an image attract visual attention. The first models concentrated on modelling gaze behavior using low-level features, such as color, intensity and orientation (Treisman and Gelade 1980; Koch and Ullman 1985; Itti et al. 1998). Such models compute for each pixel of an image a measure of saliency, the result of which is called a *saliency map*. However, it has been shown that task-related gaze behavior can dominate over saliency (Land et al. 1999). Per-pixel measures of task relevance have more recently appeared, and these are called *task maps* (Cater et al. 2002; Navalpakkam and Itti 2005; Sundstedt 2007). Established theories about visual search assume that low-level features characteristic of target objects (e.g. color or intensity) are enhanced and guide the search (Wolfe 1994). A similar intuitive interpretation is that top-down control raises the saliency of important objects (Oliva

et al. 2003; Navalpakkam and Itti 2005; Elazary and Itti 2008). However, though there are reasonable theories for top-down mechanisms concerning visual search tasks, they do not directly explain how attention is deployed in complex and changing tasks, such as those occurring in computer games. Eye-tracking studies are a reasonable method for investigating top-down attention from the opposite perspective, that is, analyzing how visual attention behaves under particular complex stimuli and tasks. The following section will describe eye tracking methodology in more detail and how it has been used in relation to computer games.

25.3 Eye-Tracking Methodology and Games

Eye-tracking is a technology developed to monitor eye movements allowing us to determine where an observer is looking at a given time. An eye tracker is used to sample the state of the human eyes. For each sample, a *gaze point* (a 2D location in screen space) is estimated. This information can give us insight into what attracted the attention of an observer or what they found interesting (Duchowski 2003). Eye trackers measure the physical rotations of the eyes to determine the gaze direction. Gaze has also been referred to as the vector between the eye and the gaze point (Hornof et al. 2003). This information can be recorded and used in offline analysis or for real-time interaction.

The most common system for capturing eye movements is the video-based corneal reflection eye-tracker (Duchowski 2003). The main advantage with this method is that it can be non-intrusive and does not necessarily require the user to wear anything. In video-based eye tracking, a camera is focusing on one or both eyes while the eye movements are being recorded. The light source reflection on the cornea (caused by infrared light) is measured relative to the location of the pupil's center. These two points are used as reference to compensate for head movements. This is the way it works for remotely installed light sources.

Before operating a video-based eye tracker, a calibration process is necessary to fine-tune it for each individual user (Poole and Ball 2005). A common calibration method is to measure gaze at predefined strategically positioned stimuli on screen, such as the corners of a grid (Duchowski 2003). Eye trackers normally produce a large amount of raw data since humans perform several saccades per second. A typical gaze data sample includes for each eye a 2D gaze point, the pupil's 2D location in the camera image, the distance of the eye from the camera, the pupil's size, a timestamp in milliseconds and a unique sample identification number (Tobii 2006). For more information regarding gaze data samples, please see the overview by Ramloll et al. (2004). The raw data needs to be filtered and reduced before it can be analyzed. In this process it is common to identify fixations and saccades (Rothkopf et al. 2004). Sometimes blinks are also identified as separate events. The identification of fixations is a complex problem and there is no unique method for filtering the raw data (Salvucci et al. 2000; Hansen and Ji 2010).

Eye-tracking is often used under the assumption that there is a strong correlation between the focus of gaze and the actual focus of visual attention. Indeed it is

possible to focus mentally on stimuli in the peripheral visual field, outside the foveal region. In this case, the internal visual-attention system (covert visual attention) is focused on a particular place, whereas eye-movements (overt visual attention) are directed to other places. For many applications, such as rendering 3D environments, the prediction of overt attention may be sufficient to perceptually optimize rendering of specific objects since regions outside the fovea are not perceived in high detail. In such cases the focus of attention may be estimated by a predictor algorithm, while eye-tracking is used only to infer the predictor in the first place and to evaluate the performance of predictor heuristics (Marmitt and Duchowski 2002; Peters and Itti 2008).

Jacob and Karn (2003) give a comprehensive overview of eye tracking in human-computer interaction research. This is a review of work regarding the application of eye movements to user interfaces both for analyzing them (usability measurement) and as a control mechanism (input). Jacob and Karn summarise a range of usability studies and discuss what users, tasks, and eye tracking metrics were used. Some mentioned eye tracking metrics include fixations, gaze duration, area of interest, scan path, etc. In addition to estimating the position of the foveal focus, various other features useful for analysis, such as fixation counts or amplitudes of saccades, can be extracted from eye-tracking data (Duchowski 2003).

Wooding and David (2002) introduced the concept of fixation maps as a means of quantifying eye-movement traces. Wooding also explored the concept of similarity between eye-movement patterns from different individuals and to which degree their fixations covered the image. Overlapping fixations are visualised using a three-dimensional surface plot, also referred to as a landscape or terrain based on the fact that the value of any point indicates the height or amount of property (discrimination/detection/perception) at that point. Wooding pointed out that the fixation duration can be taken into account by creating a dwell map, which also represents not only the areas fixated, but also the time these were fixated upon. Notably fixation duration, which is used in this chapter to weigh fixation counts, is suggested as a good indicator for estimating how strongly cognitive functions, such as object identification (De Graef et al. 1990), memory (Henderson et al. 1999) and monitoring of task-relevant objects (Land et al. 1999) are involved. The relationship between human gaze control and cognitive behavior in real-world scene perception is reviewed in (Henderson 2003).

There are various application areas, including computer graphics, virtual reality, and games, where saliency and task models have been used with varying degrees of success. In graphics for example, these models have been used to inform global illumination algorithms (Yee et al. 2001; Haber et al. 2001; Cater et al. 2003; Sundstedt et al. 2007). Luebke et al. (2000) and Murphy and Duchowski (2001) demonstrated that geometric detail in the periphery of the visual focus can be reduced without decreasing the perceived rendering quality by using an eye-tracker for gaze-contingent rendering optimizations. Komogortsev and Khan attempted to predict the visual focus of multiple eye-tracked viewers in order to perform perceptually optimized video and 3D stream compression (Komogortsev and Khan 2006). Gaze behavior was also studied when certain tasks had to be carried out. To analyze

gaze behavior in natural tasks, several studies were conducted with easy tasks ranging from handwashing to sandwich-making (Hayhoe et al. 2003; Canosa et al. 2003; Pelz and Canosa 2001).

There are different ways of analyzing eye tracking data stemming from computer game players. First, the game can be played and eye tracked in real-time, storing the relevant information for later analysis. The second option is to show pre-recorded videos from the game, but then the observer is only passively observing the game and does not interact with the application, which might affect the eye movements. Finally, eye tracking could be used to analyze screenshots/still images from a game. The first option is the most flexible since it allows for player interaction and a more natural gaming scenario.

Recent studies suggest that in adventure games, fixation behavior can follow both bottom-up and top-down processes (El-Nasr and Yan 2006). Visual stimuli are reported to be more relevant when located near objects that fit players' top-down visual search goals. In first-person shooter games, as opposed to adventure games, gaze tends to be more focused on the center of the screen (Kenny et al. 2005; El-Nasr and Yan 2006; Bernhard et al. 2010). In an experiment involving active video game play, nine low-level heuristics were compared to gaze behavior collected using eye tracking (Peters and Itti 2008). This study showed that these heuristics performed above chance, and that motion alone was the best predictor. This was followed by flicker and full saliency (color, intensity, orientation, flicker, and motion). Nonetheless, these results can be improved further by incorporating a measure of task relevance, which could be obtained by training a neural network on eye tracking data matched to specific image features (Peters and Itti 2008).

Starker and Bolt proposed using an eye-tracker to guide synthesis of speech in a way that narration refers to the current object of the user's interest (Starker et al. 1990). Although eye-tracking is used for real-time user-to-system feedback, their models of interest map gaze to objects, and successively the user's level of interest for each object is inferred. This resembles our methodology of inferring objects' importance by mapping eye-tracking data to semantic properties. In recent years, an increased number of eye-tracking experiments have been conducted using virtual environments or computer games (Rothkopf et al. 2007; Kenny et al. 2005; El-Nasr and Yan 2006; Jie et al. 2007; Sundstedt et al. 2008). These studies support the hypothesis that in conditions where a task has to be carried out, gaze behavior is mainly dominated by task relevance rather than salient features in the stimuli, as task-relevant objects are continuously monitored by the visual system (Land et al. 1999). Note that once a target is found and monitored during a task, the models for top-down control from visual search are no longer appropriate.

In the last few years, there has been an increasing amount of work done in the field of studying visual attention in games and using gaze to control games. Sundstedt (2010) and Isokoski et al. (2009) give more extensive overviews of visual attention studies in gaming and the use of eye tracking as an interaction device. El-Nasr and Yan, for example, studied the differences between players' eye movement patterns in two 3D video games (El-Nasr and Yan 2006), assessing whether the eye movement patterns in a game follow top-down or bottom-up processes. They found

that exploiting visual attention in games can help reduce frustration and increase engagement (El-Nasr and Yan 2006). Kenny et al. presented a study which investigated eye gaze data during a first-person shooter (FPS) game in order to find which information was more important in distributing interactive media algorithms (Kenny et al. 2005). Sennersten studied eye movements in an action game tutorial and was interested in how players direct their gaze, in particular what, where and when they fixate on specific objects (Sennersten 2004). Sennersten and Lindley (2009) used a real-time gaze object logging system to investigate visual attention in an FPS game.

McDonnell et al. studied a variety of humans in crowds to determine which parts of the characters people tend to observe most (McDonnell et al. 2009). Jie and Clark developed a 2D game in which the strategy and difficulty level was controlled based on the eye movements of the player (Jie et al. 2007). Hillaire et al. developed an algorithm to simulate depth-of-field blur for first-person navigation in virtual environments (Hillaire et al. 2008). Later, they also used a model of visual attention to improve gaze tracking systems in interactive 3D applications (Hillaire et al. 2010).

25.4 Understanding Playing Behavior Based on Eye Tracking

Understanding player behavior is important for both game designers and researchers, for instance in evaluating player experience. There exist several ways to analyze such behavior, each potentially revealing different aspects of the psychology involved in playing computer games. This chapter focusses exclusively on one such approach, namely the study of eye-tracking data obtained while participants are playing games (Sundstedt 2007, 2008; Stellmach 2007; Nacke et al. 2008; Bernhard et al. 2010; Bernhard et al. 2011).

Gaze analysis on the basis of eye-tracking data (Ramloll et al. 2004) yields fine-grained information regarding objects and events that are typically attended to in games (Sundstedt 2007; Stellmach 2009). We see this as a valuable tool that can be employed during the design cycle of novel games, as it can reveal where players are focussing their attention.

One of the earlier experiments in this realm maps fixation points to objects in a pseudo-3D game scenario, which is then used to answer the question as to whether the presence or absence of a task influences fixation behavior (Sundstedt 2007; Sundstedt et al. 2008). They record the full game state, enabling the game engine to later replay all actions, thereby facilitating the mapping of fixation points to potentially moving objects. Later, it was shown that this approach extends to full dynamic 3D scenes (Bernhard et al. 2010), confirming the utility of gaze-to-object mapping techniques.

The findings of Sennersten and Lindley provide further corroboration (Sennersten and Lindley 2008), showing that analyzing gaze in terms of Volumes of Interest (VOIs) or Objects of Interest (OOIs) provides insights that are difficult to obtain with screen-space techniques only. Sennersten and Lindley (2008) integrate the HiFi game engine with an eye tracker to map gaze coordinates to objects in a scene. As mentioned previously, Sennersten and Lindley (Sennersten and Lindley 2009)

used a gaze object logging system to investigate visual attention in game environments.

Stellmach et al. (2010c) discuss the trends and requirements for gaze visualization techniques. They describe a user study with experts in the field and outline which features are desirable for the visualization of gaze data. One of their suggestions is to aggregate different visualizations. Specifically, they describe three gaze visualization techniques for superimposing aggregated fixation data over 3D stimuli: (a) projected, (b) object-based, and (c) surface-based attentional maps. These are briefly elaborated here.

Projected attentional maps are 2D planar overview representations of 3D gaze data. They can be informed by a 2D Gaussian distribution in a manner similar to contour plots (Wooding and David 2002; Stellmach et al. 2010c). If the view changes, the projected attentional maps have to be recalculated. The performance does not depend on the size of the scene or the number of objects, and it is accelerated with less gaze data.

The *object-based* approach assigns a color value to each 3D object to describe its visual attractiveness (e.g. visual attention). The performance of this method is independent of the viewpoint and is only affected by object and gaze data quantities. This bears similarities to the aforementioned gaze-to-object mapping techniques.

The *surface-based* approach displays gaze data as 3D heat maps on the surfaces of the model. A gaze ray is mapped to the triangles of the mesh and a 3D Gaussian is used to splat gaze information across the mesh surface. Here, the mesh needs to be carefully chosen to obtain smooth attentional maps (Stellmach et al. 2010b). The surface-based attentional maps are the most time consuming to compute and performance is affected by the amount of gaze data and model complexity. Similar to the object-based approach, it is also independent of viewpoint modifications.

The main contribution of Stellmach et al. (2010b) is toward better visualization techniques for 3D stimuli via the three mentioned attentional maps, albeit without the goal of improving gaze-to-object mapping techniques. These 3D attentional maps aim to assist in visually better comprehending and inspecting the various aspects of gaze data (e.g. fixations duration, count, frequency). They may also be combined to provide visualizations at different levels of detail. The work was conducted with a static 3D scene and a dynamic viewpoint, but does not include dynamic objects.

Stellmach et al. (2010a) extend upon this work by experimenting with 3D scan path visualizations. They also introduce the models of interest (MOI) timeline, which can help to determine which object was viewed at a specific instant, thereby serving the same purpose as the recording of game states (Sundstedt 2007; Sundstedt et al. 2008). Additionally, they visualize the camera path with traces pointing at each gaze position.

Alternatives to the study of eye-tracking data are also in active development. We see these alternatives as complementary sources of input. For instance, in-game events may be logged for the purpose of analysing the nature and frequency of events occurring during game-play (Nacke et al. 2008; Nacke et al. 2011; Sasse 2008), although this approach does not take into account the physiological responses from the player. Nacke et al. (2011) present a logging and interaction framework

(LAIF) which enables those inexperienced with game design and programming to develop games and analyze them in a research environment. The work includes a user study based around a 2D gaze-interaction game which is playable with mouse input. Sasse (2008) gives an extensive overview of logging techniques for games as well as presents further information regarding the game used in the LAIF framework (Nacke et al. 2011).

Finally, questionnaires could be used to gather additional information, focussing on the emotional state of the player. Nacke et al. (2008) present a psychophysiological logging framework (Stellmach 2007) and discuss how the input from such a system can be synchronized with automatic scoring of game events. Questionnaires are useful for assessing the extent of spatial presence as well as gameplay experience (Nacke et al. 2011).

25.5 Overview of a Practical Pipeline to Measure Gaze Behavior in Games

Understanding what a game player is looking at during gameplay may help improve the design of a game. However, traditional tools and techniques, typically screen recording and playback, give very limited information. Matching fixation points to *pixel* data would allow us to understand game play in terms of low level features such as pixel color and contrast. To go beyond that, a number of different approaches that focus on mapping gaze to the underlying *objects* that give rise to the observed visuals have recently emerged (Sundstedt et al. 2008; Sennersten et al. 2008; Stellmach 2009). This mapping can be performed at different levels using various algorithms, as discussed in Sect. 25.6, but it does not reveal any information regarding the semantics of game play. In this section, we will describe the principles of designing a pipeline that enables not just correlating gaze to geometric objects, but also going a step further, allowing to map gaze to semantic objects, thus affording the opportunity to learn how users interact with games. An overview of the generic pipeline described in this section is shown in Fig. 25.3.

25.5.1 *Adapting Games to Study Gaze Behaviour*

To allow gaze data to be mapped to geometry, the game needs to be designed in a specific manner. When designing games, it is standard practice to use structured and systematic methods of naming, categorizing and grouping content. For example, in an FPS game, a category of “enemies” may be used to group together different classes of enemy types. An enemy class may be “soldier” or “aircraft”. Furthermore, multiple individual instances of enemies belonging to the same class are commonplace in a game (e.g. “soldier_20” or “aircraft_12”). Similarly, game content can be enriched with other useful properties such as object color or shape features (e.g.

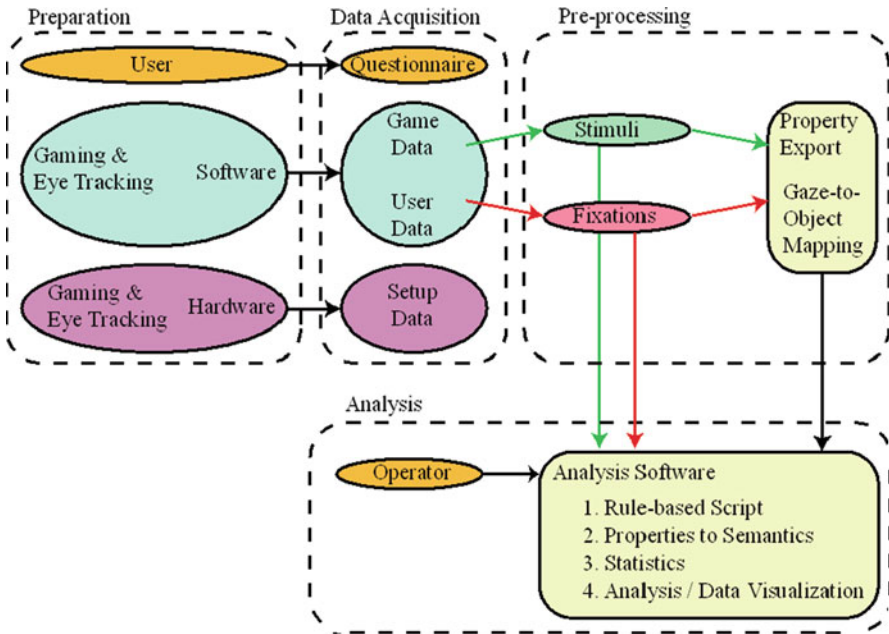


Fig. 25.3 Overview of the generic pipeline described

round, flat, etc.), status (e.g. a door may be open or closed, an enemy unit may be dead or “activated”). This wealth of information present in the game content itself provides a higher level description of the game and can be captured and processed to infer meaning of the user’s gaze behavior later.

Further, games should be modified to provide game recording and playback functionality. Recording is responsible for capturing and storing the state and characteristics of the entire game for later offline use. To implement recording functionality, the game’s scenegraph can be traversed and sampled at discrete time intervals. The best choice is to use the rendering loop of the game and capture the desired parameters whenever a new frame is rendered and dispatched to the screen. In turn, the playback functionality will then be able to use a recorded gaming session to load all the game state parameters necessary into the game engine and reconstruct the stimuli at a particular sample. As computing fixation points is an offline process, this functionality will allow us to determine which objects were attended to.

25.5.2 Preparation

Preparing for an eye-tracked gaming session is similar in principle to standard eye tracking studies (Sundstedt et al. 2009). However, games can be computationally demanding, thus requiring better hardware than that used when eye tracking simpler

applications such as web browsing or office applications. Most eye trackers can be operated remotely, allowing one workstation to be used exclusively for eye tracking, while a second computer is used for handling the game. Note however, that special attention must be paid when synchronizing the eye tracking data and the recorded stimuli, since clocks between two computers are unlikely to be in sync. Selecting a non-obtrusive (e.g. not worn by the player) eye tracker enables users to focus on the game itself, in contrast to head-mounted devices.

The higher the sampling rate an eye tracker can achieve the better, since fast eye movements can be more accurately recorded. It is wise to use an eye tracker that samples eye gaze at an equal or higher frequency than the display's refresh rate. This allows us to have more than one gaze sample per frame, making gaze-to-stimuli correlation more robust over temporal windows. Some commercial eye trackers currently offer sampling rates that exceed 100 Hz, while the majority of standard LCD displays operate at 60–75 Hz in native resolutions. Lighting conditions of the viewing environment should remain constant. Eye tracking data quality can be improved by the use of a chin rest, used to stabilize the head, however in most game-related studies the use of a chin rest is discouraged, because this alters the natural behavior of game players, which usually involves changes in body posture and head position in relation to the screen. Any instructions should be provided to the player a priori.

Finally, it is important to test the eye tracking hardware and be aware of the limitations it may have. For example, in some eye tracking devices, data quality degrades as the gaze moves away from the center of the screen and toward the corners. Also attention should be paid to participants wearing eye-correction glasses, occluding eye lashes and eye lids, lazy eyes, small and large pupils or pupils with low contrast. In addition, gaze behavior of participants may be affected if the setup does not resemble that of a natural gaming situation. When performing studies, care should be taken that participants may presume a certain task in computer games, even if none is provided, as this could alter their gaze patterns. Finally, participants should not partake in the same experiment more than once to avoid learning effects.

25.5.3 Data Acquisition

When studying gaze behavior in computer games, the data channels worth capturing are determined largely by the type of analysis to be performed later. Although there is no standard, there are four categories of data that one should consider recording:

1. *Setup data*: the characteristics of the environment and hardware used (e.g. eye tracker and screen), as well as its parameters (e.g. sampling rate and screen size), are important for later analysis. Parameters that belong to this category are static; that is, they remain the same throughout a study and across different subjects.
2. *User data*: in this category belong data referring to or produced by the user. This Calibration data can be both static (age and gender) and dynamic.

3. *Calibration data*: the latter includes calibration data, eye gaze over time (e.g. time-stamp, position gazed in 2D screen coordinates, blinks, etc.) User input via the game's controlling interface, while technically acquired through the game engine, can also be conceptually classified as belonging to this category.
4. *Game data*: this category encompasses data produced by the game. In the past, screen recordings have been the primary game data acquired, however, as explained in this chapter, this is very restrictive. Instead, in gaming environments there is a wealth of information to our disposal, not only about the stimuli shown to the user, but also the parameters used to arrive at them, as well as temporal aspects and intrinsic parameters of the game's state. The range of data types available via the game's scenegraph is very wide, and game content can be further enriched by its designers to include properties and states. In most studies, the camera parameters, the game entities' parameters (e.g. position, orientation, color, textures, etc.) and game-generated events are the best candidate data types for capturing. Apart from dynamically changing data, games also have static data that can be recorded once, for instance the window size and position, which may differ from those of the screen, the hardware it runs on, etc.

The purpose of acquiring these data is to be able to reliably and accurately reconstruct the stimuli that affected the user's gaze behavior with the goal to study it. This decoupling of data acquisition and data analysis provides an ideal methodological partitioning that allows data reuse. The data captured from all these categories has very low storage requirements even for several minutes of data acquisition. Notably, game data consists only of parameters that allow the reconstruction of the game state at any given time, without the need of capturing thousands of images, as is the case for a screen recording.

Finally, it would be advantageous to debrief participants by means of a questionnaire. This could serve two purposes. First, a well-designed questionnaire would make it evident whether the participant understood the task that was being performed. Testing this can be important, because participants that have either misinterpreted the instructions or have second-guessed the purpose of the experiment may yield unreliable or biased data. In essence, if an outlier is detected by analysis of the data, then the questionnaire may help explain why this has occurred, providing the justification for outlier removal.

Second, the questionnaire could contain questions that query the participant regarding their response to the experiment. For instance, it would be possible to ask how difficult the different conditions were to the participant, or to what extent the task was enjoyable. Dependent on the primary aim of the experiment, answers to such questions may corroborate the data found in the main experiment. Nacke et al. (2009) studied navigation using gaze as input in a 3D first person shooter game. The purpose of the study was to investigate the gameplay experience using gaze interaction by the use of subjective questionnaires. Three questionnaires were used based on previous work which evaluated the self-reported game experience, flow, and presence.

25.5.4 *Reconstruction and Pre-processing*

The next step in this pipeline deals with the reconstruction of the stimuli and pre-processing of the recorded data for further analysis. With the recorded data in hand, a playback-like simulation of the game is performed by reconstructing all states the game has gone through during the gaming session on a frame-by-frame basis. The data is processed off-line without any performance constraints. Several tasks are carried out:

- **Fixation detection.** Raw gaze data captured by the eye tracking hardware are fuzzy and should not be directly used to infer a subject's gaze behavior. Instead, raw gaze data is processed using fixation detection algorithms that cluster raw gaze into fixations for subsequent use (Duchowski 2003).
- **Gaze-to-object mapping.** Gaze is correlated with objects. Here, detected fixations are used to map gaze data back to scene objects, using a so-called gaze-to-object mapping algorithm (see Sect. 25.6). In this process, each fixation is in turn correlated to one or more objects which are potentially the targets of that fixation.
- **Property exporting.** Game entities carry properties assigned to them at design time. These may be static or change in the course of the gaming session. They should be linked to the objects to determine the semantics of each object.

The output of the pre-processing can be stored in a single file (e.g. in XML format) comprising an entry for each fixation. Each fixation entry contains the ID of the fixation target object(s) and a sequence of frame entries, encoding the states of the stimuli within the duration of a fixation. Each frame entry comprises a set of visible objects, a set of audible sounds, a set of user events and further attributes reflecting those properties of the game's context which may be relevant for understanding the behavior of the player. The entry corresponding to each object, sound event etc., comprise an identification code (ID) and a set of properties which characterize meaning and state of the respective entity. Correct alignment in the time domain between the recorded gaze points and game-states is achieved by comparing time-stamps of both sequences.

25.5.5 *Analysis Tools*

The final step of the pipeline is the analysis of gaze data, for which we describe our approach here. We first perform an explorative analysis by means of visual interpretation of gaze density histograms. The main analysis will then map gaze data to objects with a set of pre-defined semantic properties. The user has many degrees of freedom in the definition of the semantic properties of interest, a task aided by a user interface that allows not only the selection of semantic properties, but also enables the definition of clusters of semantic properties (e.g. assigning objects of a similar category to one super-class) or to define new semantic properties which depend on

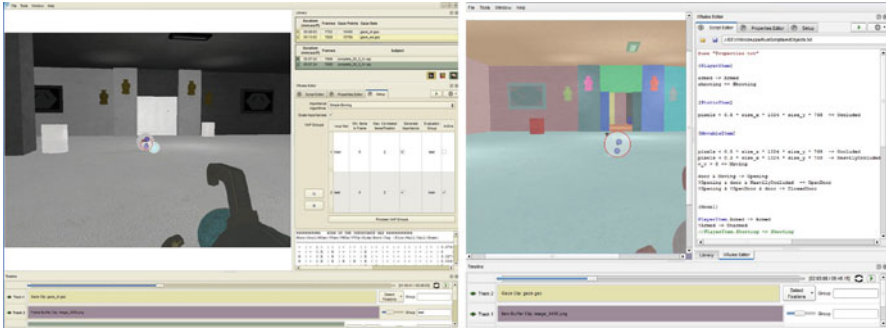


Fig. 25.4 Screenshots of the analysis software toolbox of the experimental pipeline

dynamic events in the scene (e.g. an enemy is destroyed after a successful shot). To this end, a scripting language is used in the analysis software, where rules can be defined to describe how certain properties should be interpreted prior to building gaze histograms (see Sect. 25.7).

The Graphical software tools we designed for studying gaze behavior are similar in look-and-feel to video editing suites and includes the following components:

- **Library.** A library widget designed as a front-end to gaze and stimuli data which the user can load from disk. This is useful so that the operator can select collectively which stimuli data and eye tracked subjects are relevant to his current analysis. The library holds pointers to data, but need not load the data.
- **Timeline.** A timeline widget with different stacked tracks allows the operator to instantiate stimuli and gaze data so that they can be played back in parallel. The timeline offers typical controls of temporal data (e.g. start, stop, etc.) and allows for seeking arbitrary frames within the datasets, enabling intuitive non-sequential access to them.
- **Views.** To visualize the data, viewing widgets use the timeline tracks to sequentially overlay visual representations of the respective data at the time the timeline's head is positioned or a temporal window around it. For example, fixations can be easily overlaid and played back over the stimulus that produced it.
- **Script editor.** A scripting editor enables an operator to define, execute and debug scripts that transform low level extracted game entity properties into semantic properties.

The combination of these tools into a single graphical user interface enables an operator of the analysis software to potentially gain insight and assist him in scripting rules for transforming properties to semantics. This graphical user interface, shown in Fig. 25.4, is effectively an Integrated Development Environment (IDE) for studying gaze behavior that not only offers the tools to setup and perform an analysis task, but also provides visual feedback and can potentially leverage the experience and intuition of the operator.

The following section describes the various algorithms required to implement such a pipeline, including those that enable the analysis of data.

25.6 Object-Based Gaze Analysis Algorithms

The input assumed for the processing pipeline is a gaze data set from the eye tracker and a reconstruction of the game states. The goal is now to process gaze data and obtain gaze statistics scoring the amount of attention deployed to particular objects, object categories or semantic properties. To achieve this goal, two basic steps are required: (a) gaze has to be *mapped* to objects, which can then be further abstracted by their category, other properties or their meaning to the user, and (b) a method has to be defined to build gaze *statistics* with respect to the independent variables we are interested in (e.g. object IDs, properties or meaning; see Sect. 25.7). Initial solutions were prototyped in the work of Sundstedt et al. (2008) and Bernhard et al. (2010). In the following sections, the ideas behind these approaches are presented.

First, an important step of the proposed methodology is to map gaze data to objects. This is done by a gaze-to-object mapping algorithm which specifies the potential target(s) of each fixation. Fixation targets are individual objects which are represented by an identification number (ID). In some cases, it might be interesting to quantify how often an individual object was attended, but for realistic game levels we have to assume that each player has a unique game experience when navigating a spatially large environment containing many objects. Under these circumstances, gaze is distributed very sparsely and is not suited for a statistical analysis. Therefore, rather than focusing on particular object instances (e.g. “AlienMonster_57”) it is more promising to compute gaze statistics for object categories or semantics.

Overall, gaze analysis can be performed at different levels of abstraction. We distinguish four layers in which the stimulus can be represented in the analysis:

- **Screen space:** Gaze points in 2D (e.g., position=[0.1,0.5])
- **Object space:** Object instances (e.g., ID=2,933)
- **Property space:** An object’s category, state and behavior (e.g. category=“Alien Monster”, distance=5 m, behavior=“approaching”, avatar health state=10%, etc.)
- **Semantics:** An object’s meaning to the user according to game task (e.g. “attacker”, “close”, “dangerous”, “high risk”)

In Fig. 25.5, we illustrated the levels of abstraction with an example of a game where a user has to move a pedestrian across the street: In the first abstraction layer (top), we see pixels as seen by the player of the game. The next abstraction is the object level, where we have particular instances, such as cars and trees, with unique IDs. In the third layer, individual objects are abstracted in terms of their properties including the object category (e.g., “car”) and spatial properties (e.g. velocity or position). In the semantic layer (bottom), the scene is abstracted according to the meaning of the objects to the user and the task at hand. In this example, the user has to move the avatar across the street, the avatar hence becoming a pedestrian. For a pedestrian, objects which are most task relevant are the oncoming car and the car currently passing, whereas the car which has already passed by is not important. On the other hand, details of objects behind the street (e.g., houses, trees and sky) are of low relevance and can be abstracted as background.

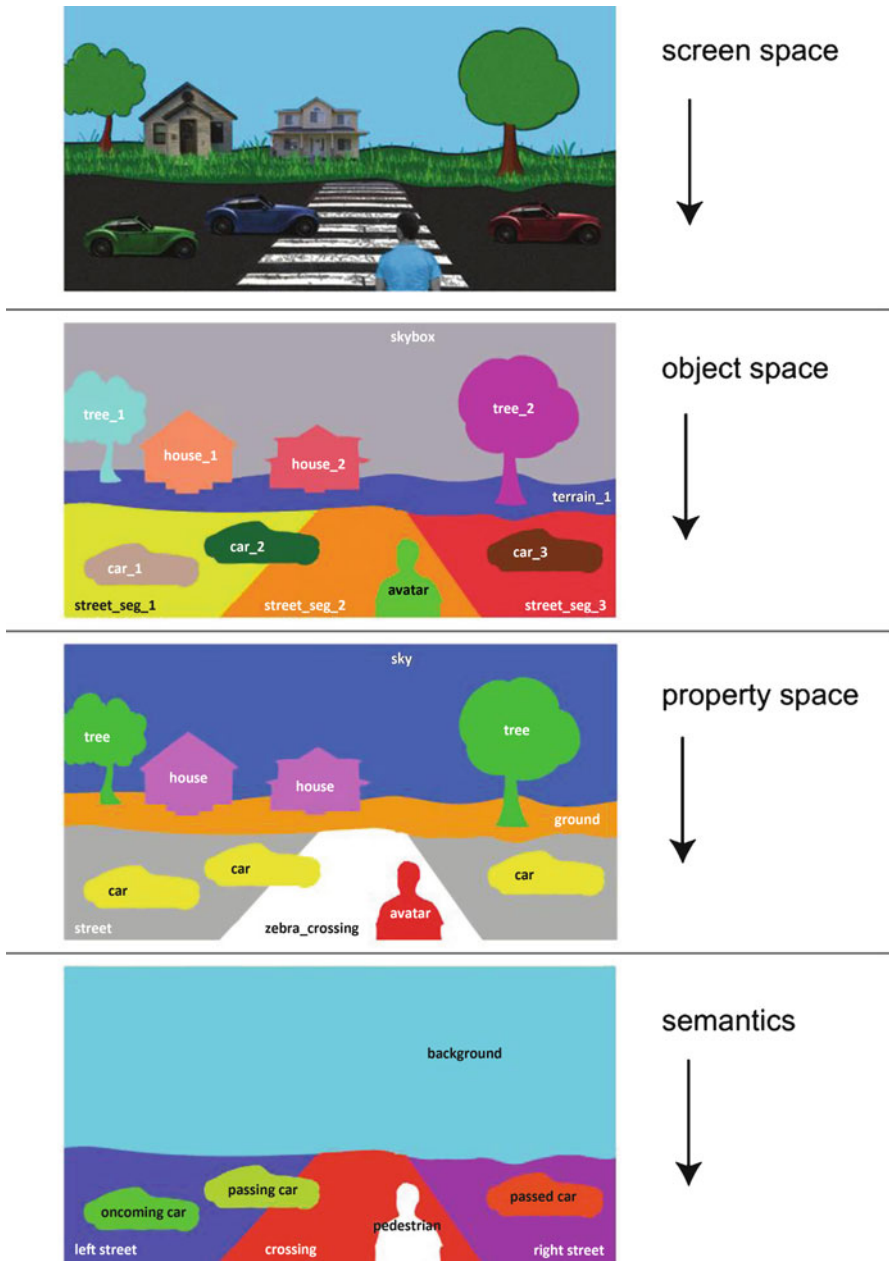


Fig. 25.5 Example for layers of abstraction in a pedestrian road crossing task

Most commercial gaze analysis tools operate only with screen space data which is readily available: the images rendered to the screen and the gaze data which is output by the eye tracker in screen space. But for computer games, we can

fortunately assume that an object space representation is available, which can be obtained when the internals of the game engine can be accessed.

However, for gaze analysis in object space we also need a representation of gaze data in object space (e.g., the ID of a fixated object). This can be obtained by mapping gaze to objects, as described in Sect. 25.6.1. Some additional modifications of the game engine are also needed to derive properties from object space. These will be discussed in Sect. 25.6.2. Semantics are then derived from the properties of a scene. Since inferring semantics from properties is a cognitive process, this requires the assistance of a human operator who provides an ontology which defines the mapping from properties to semantics (Sect. 25.6.3).

25.6.1 *Mapping Gaze to Object Space*

A common way to pre-process gaze data is to filter for fixations, since a user's attention correlates with fixation locations only and not the saccades in between (Duchowski 2003). Thus, the input of gaze-to-object mapping is assumed to be fixations and the reconstructed states of the game during the start and end time of each fixation.

Note that the position of a fixation is fuzzy, as it corresponds to a cluster of jittered gaze points sampled by the eye tracker during the time the fixation occurred. On the other hand, we have objects of a 3D scene which are rendered to a 2D image by a perspective projection from the camera viewpoint. Mapping fixations to objects is therefore done by computing the degree of intersection between fixations and scene objects. Different ways to achieve this have been proposed, reaching from straight-forward solutions to more sophisticated methods. The methods along with their advantages and disadvantages are briefly described in the following (see also Fig. 25.6).

25.6.1.1 **Point-Based Methods**

The simplest way to map fixations back to objects in a scene is to use the center of a fixation (e.g. the mean position of raw gaze samples). With this simplification, one has to find only the object which was projected to one pixel position. This can be carried out directly in object space by casting a ray through the fixation center into the scene and computing the nearest object intersected by that ray.

Another solution would be to solve the problem in screen space. Each scene object is rendered using a unique color ID and stored in an image buffer, referred to in the literature as an *item buffer* (Sundstedt et al. 2008) or *id buffer* (Saito and Takahashi 1990). This operation can be performed directly on the GPU with minimal computational effort. The color of the pixel which corresponds to the fixation position is then queried in the item buffer and decoded to obtain the respective object ID.

Using only one point to map a fixation to an object is a reasonably simple and efficient solution, which may be advantageous for real-time applications. It should

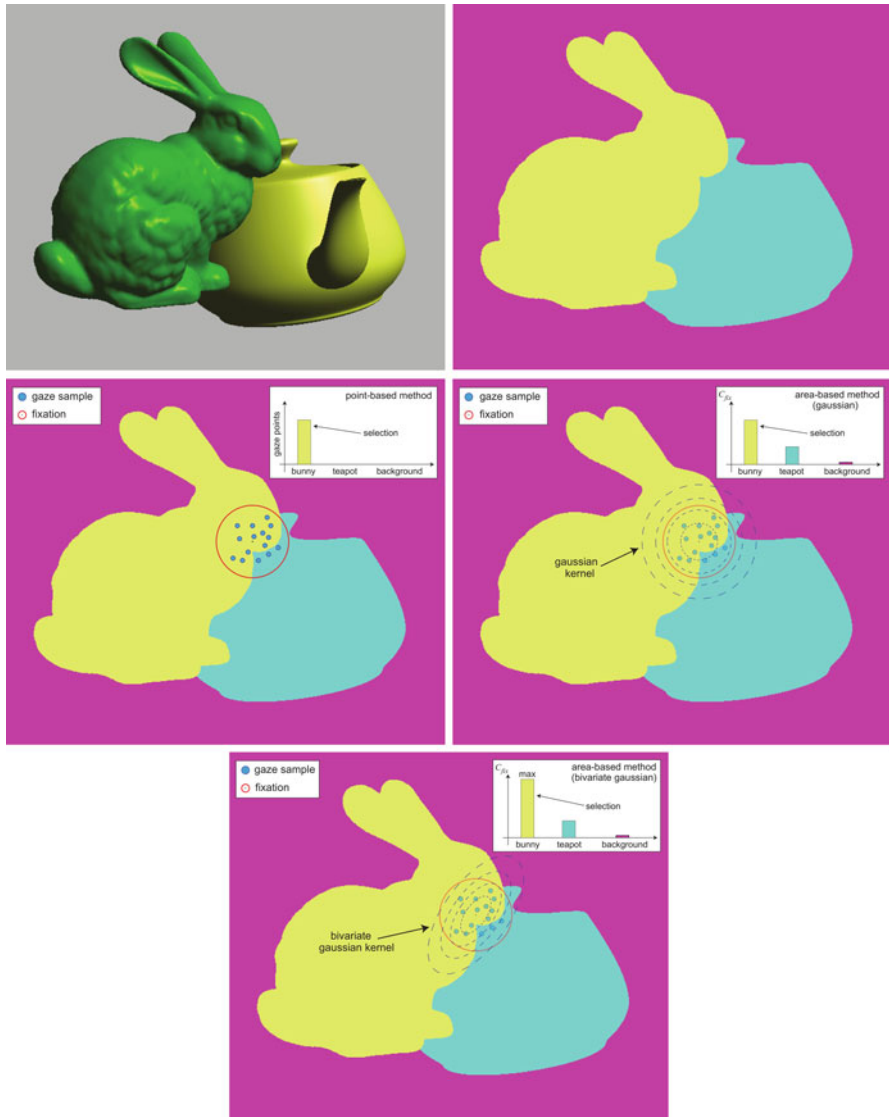


Fig. 25.6 Gaze to object mapping methods: to identify the object underneath each pixel, the scene (*top left*) is rendered to an item buffer (*top right*). Fixations can be mapped to objects by simply picking the pixel at the center of a fixation (*middle left*) or integrating the energy spread by a 2D Gaussian kernel (illustrated with *dashed rings*), which models the foveal acuity (Eriksen and St James 1986) (*middle right*) or the distribution of gaze points (Bernhard et al. 2011) (*bottom*), over the area of the respective objects

work well as long the scene is simple, that is if there are only few, relatively big and well separated objects. But in many cases, the fixation center might not necessarily intersect the object a user is actually attending.

There are two factors that may play an important role when mapping gaze to objects: first, the fovea resolves all objects in sharp resolution in a frustum of about 2° of visual angle (Palmer 1999), and second, a fixation is a cluster of a fuzzy cloud of gaze-points distributed over space and time being sampled in discrete time intervals from an actually continuous motion path of two eyes. It is, therefore, not always appropriate to evaluate the intersection of a fixation and scene objects at a single point. Rather, it may be advantageous to account for the region spanning the potential focus of attention, thus leading to area-based methods.

25.6.1.2 Area-Based Methods

An area-based approach was first proposed by Sundstedt et al. (2008). They render the scene into an item buffer and intersect a kernel K_{fix} with the objects contained in the buffer. The kernel is centered at the fixation's mean position $(\bar{x}_{fix}, \bar{y}_{fix})$ and for each visible object o , an integral is computed which accumulates the energy contributed by that kernel over the area $A(o, t)$ covered by this object in the item buffer at time t . To account for possible changes in the item buffer, integration is also done over time between start time stamp $t_{s, fix}$ and end time stamp $t_{e, fix}$. We define this integral as the correlation $C_{fix}(o)$ between a fixation fix and an object o :

$$C_{fix}(o) = \int_{t_{s, fix}}^{t_{e, fix}} \int_{(x, y) \in A(o, t)} K_{fix}(x - \bar{x}_{fix}, y - \bar{y}_{fix}) d(x, y) dt \quad (25.1)$$

Note that this implies that the fixation duration t_{fix} is given by $t_{e, fix} - t_{s, fix}$. Two variants of the kernel have been proposed so far: Sundstedt et al. (2008) proposed a kernel which simulates the visual acuity of the human retina, whereas Bernhard et al. (2010) proposed to use a kernel to model the distribution of the gaze points corresponding to the fixation.

25.6.1.3 Foveal Sensor Density Model

Sundstedt's foveal sensor density model begins by approximating the fall-off of spatial acuity from the fovea to the periphery using a normal distribution N (Sundstedt et al. 2008):

$$K_{fix}(\Delta x, \Delta y) = N(\sqrt{\Delta x^2 + \Delta y^2}) \quad (25.2)$$

The Euclidean distance $\sqrt{\Delta x^2 + \Delta y^2}$ between any pixel position and the center of a fixation point is inserted into the univariate Gaussian density distribution:

$$N(\sqrt{\Delta x^2 + \Delta y^2}) = \exp\left(-\frac{\sqrt{\Delta x^2 + \Delta y^2}}{2\sigma}\right) \quad (25.3)$$

Taking into account that the foveal region of human vision spans approximately 2° of visual angle, the area over which a fixation point bears relevance corresponds to a circle which is determined by an intersection of the cone of foveal vision and the screen plane. The size of this circle, which we denote as σ_{fovea} , is then used to define the standard deviation of the Gaussian kernel. Assuming that the distance between eyes and the display is d , we can compute σ_{fovea} :

$$\sigma_{fovea} = d \tan(\alpha) \quad (25.4)$$

Note there is a subtle but reasonable simplification in the computation σ_{fovea} as the intersection of the foveal cone and the screen plane actually depends on the eye's viewing angle and would vary with gaze position if computed accurately. However, it is more important to account for the limited accuracy of the eye-tracker. Thus, we add to this the eye-tracker error σ_{error} :

$$\sigma = \sqrt{\sigma_{fovea}^2 + \sigma_{error}^2} \quad (25.5)$$

Using Eq. (25.1), we compute a weight C_{fix} for each object, which serves as an estimate for the likelihood for it to be attended by the user. This model assumes that the a priori probability for a fixation increases with the number of pixels covered by the object. Hence, if the size of scene objects varies too much, this may result in a bias toward large objects, as the amount of attention received does not necessarily correlate linearly with size. To control large variations in size, it may be necessary to perform a subdivision of objects (as done in Sundstedt et al. 2008).

Another assumption of this approach is that several objects may be attended during one fixation, as the output is a value for each object scoring its potential attentional relevance in the current fixation. This corresponds to spatial models for attention such as the spotlight (LaBerge 1983) or zoom-lens models (Eriksen and St James 1986; Castiello and Umiltà 1990), which assume that attention is enhanced for all objects within the focus region. However, some experimental results suggest that human cognition is better at attending only one object at a time (Duncan 1984; Baylis and Driver 1993; Behrmann et al. 1998). Especially, during execution of a task, unexpected objects or events may go unnoticed even if they appear within the foveal focus of a viewer (Simons and Chabris 1999).

Under these assumptions, it is not necessarily appropriate for a single fixation to compute an attention weight for several objects. Hence, Bernhard et al. proposed to assume that during a fixation, attention is focused on one object only and may not be directly related to the foveal sensor density (Bernhard et al. 2010). Instead of approximating foveal sensor density distributions, they account for the fact that a fixation is made up of a cluster of spatially distributed gaze points, as discussed next.

25.6.1.4 Gaze-Point Distribution Model

As the eye-tracker has limited precision, and the human oculomotor system cannot hold gaze stable on a fixed position, we have to account for the fact that the gaze

points, which are sampled at discrete points in time, are distributed within a known uncertainty region (Bernhard et al. 2010). The density distribution of the continuous gaze paths during a fixation can be approximated, for instance with a bivariate Gaussian kernel. The parameters of the kernel are derived from the constant uncertainty of the eye-tracker and the spatial distribution of gaze points clustered within the fixation. A bivariate kernel provides a better fit to unidirectional drifts of gaze, which were frequently observed in the gaze data:

$$K_{fix}(\Delta x, \Delta y) = \exp\left(-\frac{1}{2(1-\rho_{fix}^2)}\left(\frac{\Delta x^2}{(\sigma_{fix}^x)^2} + \frac{\Delta y^2}{(\sigma_{fix}^y)^2} - \frac{2\rho_{fix}\Delta x\Delta y}{\sigma_{fix}^x\sigma_{fix}^y}\right)\right) \quad (25.6)$$

In this case, the parameters of the kernel depend on the distribution of gaze-points clustered with the current fixation fix . The parameters σ_{fix}^x and σ_{fix}^y denote the standard deviations of the fixation's uncertainty region in both dimensions, and ρ is their correlation in (x, y) . After evaluating Eq. (25.1) with this kernel for each object, the object which is most likely the target of the fixation is determined by selecting the object with the maximum value correlation weight C_{fix} :

$$o_{fix} = \arg \max_o C_{fix}(o) \quad (25.7)$$

It should be noted that current gaze-to-object mapping techniques are still in a premature state and their accuracy could be improved considerably, as discussed in the following section.

25.6.1.5 Limitations

First of all, the most important concern is that though these techniques provide reasonable results in proof-of-concept studies, their accuracy has not been evaluated yet. Unfortunately, evaluating accuracy for general scenes is a difficult problem, as it requires us to compare the result of the gaze-to-object mapping algorithm with the actual focus of a user. Such a comparison would require an experiment which uses other methods than eye-tracking to reliably determine which object is attended by the user.

We expect that the current algorithms fail particularly in situations where objects or the camera are moving fast. The algorithm's accuracy is also limited when objects are relatively small, consist of thin parts, are placed very close to each other or even occlude each other partially. Another problem arises when a user tends to scan the silhouette of an object, as this provides more information about its shape. In this case, unattended objects in the background may be incorrectly marked as fixated upon.

In current methods, fixations are treated as static. To account for fast motion in the scene or a moving view port of the camera, algorithms may need to incorporate the temporal dimension in the distribution of gaze samples clustered in one fixation. Therefore, appropriate gaze-to-object mapping methods should be developed for smooth pursuits, which are drifting fixations occurring when the eyes track a moving object.

25.6.2 *From Object-Space to Properties*

The so-called property space describes the properties of a scene or even the entire state of the current application. Ideally, such a description is generated for the entire scene, or at least for all visible objects. The reason why it is useful to consider, apart from the fixated object, other objects in the scene is that we should account for the context under which a fixation occurs. If the scene and the viewpoint changes, this is important to identify or note, because we need to track how many other objects could be concurring alternative targets for the fixations being issued.

There are several properties which could be of interest. Overall those can be divided into properties of objects and properties reflecting the current behavior of the user and the avatar being controlled.

25.6.2.1 Object Properties

- **Visibility:** all objects which were visible in the camera's field-of-view had a potential influence on a user's behavior and could be potential fixation targets. Visibility can be determined directly from the item buffer, as only visible objects may cover any pixels.
- **Object category:** the most important property is the category of an object, which allows us to link an object to semantics. As we reasonably assume that the category of an object is a static property, we just need a look-up-table where each object ID is mapped to the respective category of an object.
- **Spatial properties:** spatial properties, like size, motion or position in the screen space could also be of some interest in the analysis.

25.6.2.2 Player Related Properties

- **Game/player state:** the current state of the game and the player may also affect user behavior. For instance, a low health state could cause the player to focus on searching health items.
- **Interaction:** it might also be interesting to analyze gaze behavior with respect to the way the user is interacting with the application. Hence it is useful to include the actions of the avatar (e.g., "running" or "shooting") or input events from mouse, keyboard or joystick.

25.6.2.3 Logging Tool

To extract scene properties, a light-weight interface to the game-engine is defined, which is used to notify a logging tool about changes in the game's internal parameters, such as the view-matrix of the player camera or variables of scene entities (e.g., objects and other relevant of the game state). Having access to those very basic

parameters, the logging tool then computes properties which might be useful for the analysis or the inference of semantics. For example, the tool infers screen space positions, bounding windows or motion vectors from camera parameters and object world-space bounding boxes.

25.6.2.4 Log Format

Though many of the properties are static throughout the game (e.g., category), we have to assume changes in many other properties (e.g., visibility or user input events), which hence have to be logged for each frame. If only fixations are considered in the analysis, it is useful to define a format where for each fixation the fixated object and a description for all frames between the begin and end time of that fixation are logged. For each frame description, one should log the time-stamp, the IDs of the visible objects, their dynamic properties and a description of the current game state and user input events.

In XML an example for the log format could look like this:

```
<fixation>
  <duration>0.532</duration>
  <fixated object>
    <id>12423</id>
    <confidence>0.9</confidence>
  </fixated object>
  <frame>
    <timestamp>54.334</timestamp>
    <object>
      <id>12423</id>
      <visibility>1.0</visibility>
      <category>Tree</category>
      <screen_bounding_window>
        <min_x>0.1 </min_x>
        <min_y>0.6 </min_y>
        ...
      </screen_bounding_window>
      ...
    </object>
    <object>
      ...
    </object>
    ...
    <player>
      <health_state>0.7</health_state>
      <action>"running"</action>
      ...
    </player>
```

```
</frame>  
...  
</fixation>
```

25.6.3 From Properties to Semantics

Having a full description of the stimulus for each frame, it is now possible to analyze various aspects of human behavior, for instance by focussing on specific semantically meaningful object properties. Assuming top-down attention is mainly influenced by high-level processes, the most appropriate way to represent the stimulus is a description accounting for the meaning of objects according to the current task a user is performing. However, inferring meaning from object properties is a complex problem and requires introducing knowledge into the analysis pipeline. At this stage, the user of the analysis software has to specify a set of rules on how raw object properties should be translated into meaning. Therefore, Bernhard et al. proposed a simple scripting interface, which is integrated into the user interface of the analysis software.

A user may write rules defining relations, such as for instance “**palmtree is a tree**” or conditional statements, such as “**if** car.position.x < center.x **and** car.motion.x > 0 **then** car **is** approaching.” The transformation from object properties to semantics is then carried out by an interpretation unit which applies the rules specified by the user.

25.6.3.1 Keeping Degrees of Freedom Low

To avoid problems of sample size, it is important to keep the degrees of freedom low, i.e. avoid many dimensions and use a small number of semantic properties in the analysis. If there are too many semantic categories, one can reduce this number by clustering similar categories or defining semantic super-classes. The degrees of freedom can be reduced in an additional selection pass proposed in Bernhard et al.’s work (Bernhard et al. 2010). This selection pass is specified by the user of the analysis software and projects the output of the semantic transformation to those values in which the user is interested most.

25.7 Collecting Fixation Statistics

Let us assume that we have mapped each object to one semantic property to be further denoted as x . Of course, it is possible that an object has more than one semantic property, but, for simplicity, we will only assume a single property case (see Bernhard et al. 2010 for a multidimensional example). The next step is to derive a

statistic which scores the amount of attention given to each semantic property as an importance value. This statistic will be denoted as importance map $I(x)$, which links each semantic property x to an importance value. The importance ideally corresponds to the probability that an object holding x is fixated by the user.

To derive the importance map, Sundstedt et al. (2008) proposed to accumulate fixation times for each semantic property. However, in their study the viewpoint was fixed and the set of observable objects remained constant. For the general case, we have to assume a viewpoint which is not fixed and the set of objects in the camera's field-of-view may vary considerably from one frame to another. Thus, Bernhard et al. (2010) proposed a heuristic normalization strategy accounting for the different amounts of time certain objects are visible to the user.

To calculate $I(x)$, the time t_{fix} that objects with that x were fixated is first accumulated and then normalized by the accumulated time t_{vis} that objects with that x were visible during a fixation (i.e. the number of frames they were potential fixation targets):

$$I(x) = \frac{t_{fix}(x)}{t_{vis}(x)} \quad (25.8)$$

The normalization factor $t_{vis}(x)$ corrects for variations in the visibility of different semantic properties. If an object is visible in many frames but it is fixated only in a few of them, the importance value should be low, and if an object is fixated most of the time it is visible, the importance value should be high. The maximum importance value is 1 and occurs if a semantic property is fixated in every frame it is visible.

This model takes into account changes due to the visibility of objects. However, generalizing the solution to contextual dependencies is a difficult problem, as it would increase the dimensionality of the statistic to the size of all possible combinations of semantic properties, and a sufficient density of gaze samples would be difficult to acquire.

25.7.1 Limitations

Practically, it is not possible to define a normalization strategy which perfectly corrects for all latent effects resulting by the variation of the viewpoint and changes in the scene. Hence, this heuristic involves many simplifications, such as the assumption that each semantic property is perceived as one unit of attention. Defining the units of attention, which make up the number of alternative targets a user can fixate in a given view of the scene is a hard problem. For future work, it could be useful to investigate strategies which are inspired by models for pre-attentive object detection from vision research. Those could potentially allow to better quantify the amount of visual information a user perceives within the field-of-view.

Another important factor arising from the uniqueness of the game experience of each user is the variation of the contexts in which a particular object may be seen, which may also significantly influence attention. One strategy to reduce the varia-

tion is to divide large game levels into sections where a similar context can be expected and perform the analysis for those sections separately.

25.8 Results

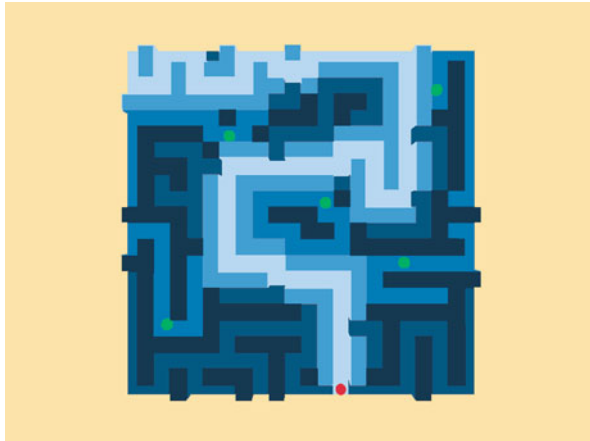
The pipeline described in this chapter is a flexible framework that offers the technical means of setting up an experiment to study gaze behavior in a game, discusses what data may be useful to capture, how to achieve this, and how to analyze them. The pipeline has been presented in a manner that can be adapted to various studies to enable researchers and practitioners to tailor it to their own needs. We will describe two sample experiments here to offer first hand examples of how the pipeline and algorithms have been put into use by the authors.

25.8.1 Example 1

The first example consists of an eye-tracking experiment that was carried out to generate an importance map based on high-level properties in a computer game (Sundstedt et al. 2008). The task of the game was to navigate a small ball through a maze, which was in 3D, but rendered from a fixed bird's-eye view. All items in the maze were tagged with high-level properties, such as the correct and incorrect path, to encode the relevance of certain parts of the maze in relation to the task of finding the exit of the maze. These items were also referred to as *object classes* and can be seen in Fig. 25.7 along with a more detailed description. Accumulating fixations over different object classes provides a fruitful approach in understanding where game players focus their attention. Such information cannot currently be extracted from an analysis of low-level salient features alone.

The analysis process depends on three main steps. In the first pass, the player plays the game while being recorded using an eye tracker. All game states are logged so that it is possible to reconstruct each frame later. The novelty of this approach is that it also allows playback of the game in real-time, which can be used for another condition or another group of players watching the same game stimuli passively, for example. After the first pass, each frame can be reconstructed and the additional data, such as the item buffer, frame buffer and object data, can be generated. Fixations can then be mapped to object types using the item buffer in order to find out which of them are the most significant with respect to the gameplay. The item buffer for the maze can be seen in Fig. 25.8. Finally the analysis tool can be used to get useful information from the stored game and gaze data to generate an importance for each object class. The distribution of fixations directly relates to the importance each object type carries for executing the game's tasks.

The area-based approach (Sect. 25.6.1.2) is used, which renders the scene into an item buffer and convolves with a kernel simulating the visual acuity of the human retina, with the objects contained in the buffer. The Foveal Sensor Density Model is used to map fixation points back to semantic object classes in the game. After











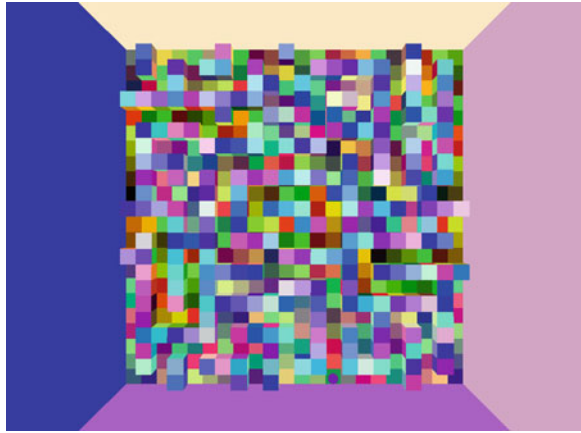
Object Class	Description
 Correct path	The floor and walls of the path that must be traversed to go from the starting point to the end point.
 Incorrect path	The floors and walls of dead ends.
 Adj. to correct	The top surface of the walls that are adjacent to the correct path.
 Top surface	The top surface of the walls that are not adjacent to the correct path.
 Closed paths	These are the parts of the maze that are separate from the main paths and may contain distracting elements.
 Main walls	The four walls enclosing the playing field.
 User-contr. ball	The ball under the participant's control.
 Distractor balls	Balls not under participant control.

Fig. 25.7 Maze Object Classes: the image shows the different object classes used in the experiment (Eriksen and St James 1986) as well as their configuration. Below this, the description of the object classes

classification, each participant produces a normalized distribution of fixations per object class. This set of distributions is then subjected to further analysis using traditional statistical tests, such as a one-way ANOVA (Cunningham and Wallraven 2011), to reveal statistical differences in fixation behavior of participants in different conditions (for example actively playing a game versus passively watching a game) (Sundstedt 2008).

Fig. 25.8 Maze Item

Buffer: showing all *color coded* objects which enable us to relate fixation points to objects and object classes



To check the validity of the experimental design, fixation distributions are matched to projected object sizes. The hypothesis is that if fixations are randomly distributed, they would fall on large objects more often than on smaller objects. It was found, however, that the fixation distributions are markedly different from the distribution one would obtain by counting the number of pixels that are covered by each object type. This indicates that none of the results can be explained by random fixation behavior.

The experimental design allows a comparison between fixation behavior while the game is played against fixation behavior while observing a recording of a previously played game. The hypothesis is that passive viewing would lead to different behavior than active gameplay. Further, if this is the case then the concept of saliency could be applied to predicting gaze behavior in the absence of a task, while simple saliency measures would not predict fixation behavior in the presence of a task.

However, this experiment led to a surprising result in that even passive user behavior is task dominated and cannot be statistically distinguished from active gameplay behavior. In this particular game design, saliency is therefore a very poor predictor for task relevance. This observation may extend to other game designs. However, it should be noted that in this study, the camera was locked so that each user had access to similar visual content at all times. This improves the rigor of the experimental design, leading to better control of the experimental set-up, and thereby fewer risks of introducing bias.

On the other hand, this study reduces the problem of inferring gaze distributions to a very limited case by assuming a fixed camera and a constant set of objects. In the second example, the approach is generalized to a representative 3D scenario with a dynamic viewpoint and a field-of-view with variable content.

25.8.2 Example 2

The second example is from Bernhard et al. (2010), who implemented an early prototype of the entire pipeline described in this chapter. A 3D First Person Shooter game was used to perform a proof-of-concept study of their system.



Fig. 25.9 Predicting visual attention from fixation statistics: (a) An example reconstructed framebuffer of the game is overlaid with the visualization of a fixation in the current frame. In Figure (b), fixation statistics were used to predict the importance for each object in the scene which is visualized by the brightness of the objects (brighter objects are more important than darker ones). Figure (c) shows the corresponding saliency map. Since fixation statistics account for semantics, they can predict better the high importance of doors or objects in the center, while saliency maps are less selective and predict the importance of pixels rather than objects

The actual goal of this work was to derive a gaze prediction heuristic which is learned from gaze data recorded from several participants of an eye-tracking study. Learning is regarded as the process of inferring an importance map (essentially a statistical model of the data) by utilizing the gaze analysis pipeline. The importance map is then used to estimate the likelihood for each object to be attended in a particular frame. Figure 25.9 shows an example frame of the FPS game, the corresponding importance map learned and the respective saliency map for comparison.

The following sections will give a very brief description of this work. For readers who prefer a more detailed description, we recommend to read the original article by Bernhard et al. (2010).

25.8.2.1 Inferring Importance Maps

The pipeline can be adapted, as shown in Fig. 25.10, to enable deriving importance maps for gaze prediction, which is structurally similar to the methodology described in Sects. 25.6 and 25.7. The input of this pipeline is a gaze file and a replay file recorded during an eye-tracking study. This information is then used twofold. First, an abstraction of the stimulus in terms of high-level semantic properties is derived. Second, the object which was fixated in that stimulus can be determined. The information as to which object is fixated and the abstraction of the corresponding stimulus then forms the input of an algorithm which learns an importance map. A straightforward estimate of the importance map can be obtained by accumulating fixation times for all semantic properties as described in Sect. 25.7.

25.8.2.2 Gaze Prediction at Runtime

At runtime, the importance map forms the basis for a per-object estimate of the probability of being attended. This probability is computed for those objects located within the field-of-view of the current frame, as illustrated in Fig. 25.11.

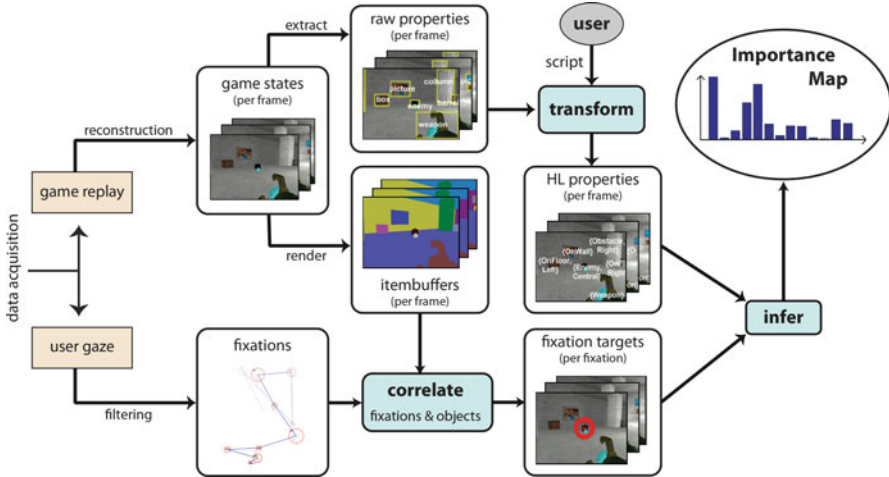


Fig. 25.10 An overview of the complete pipeline used by Bernhard et al. (2011) to derive importance maps

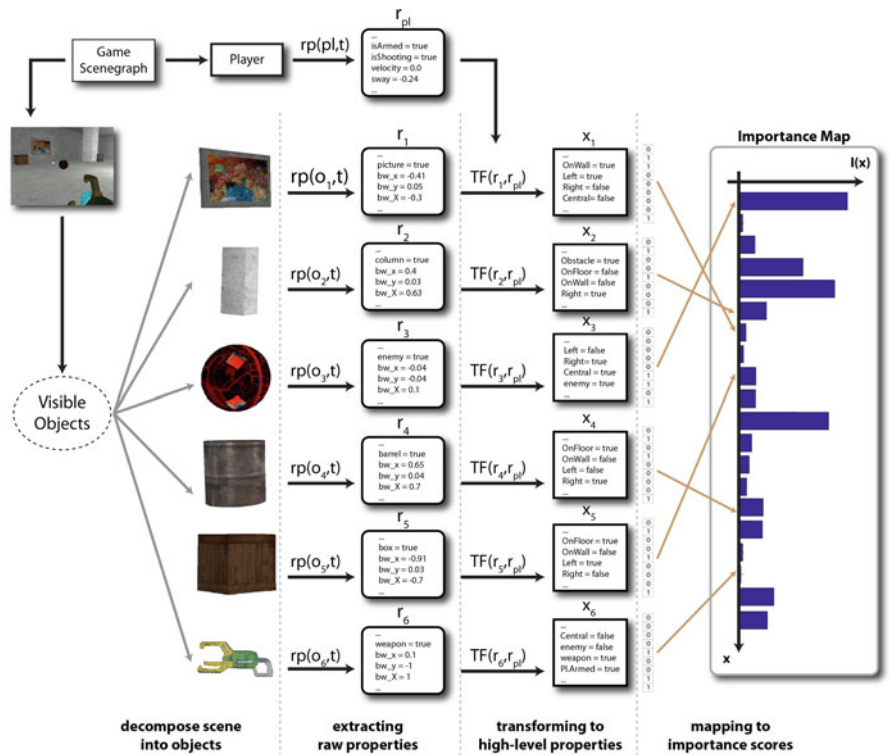


Fig. 25.11 The process of assigning importance values to objects in the player’s field of view

First, a visibility algorithm determines the set of visible objects. For each of these objects, their properties are extracted from the scenegraph (Sect. 25.6.2), which are then transformed to high-level properties with a user specified function (discussed in Sect. 25.6.3). Boolean values are used to encode whether an object exhibits a certain high-level property or not. Therefore, the output of this mapping is a vector of boolean values which are then used as keys to perform a look-up into the importance map. With this process, a normalized importance value is determined for each visible object in the scene.

25.8.2.3 Discussion

Bernhard et al. (2010) evaluated their importance maps in the context of a first person shooter game. Included in their experiments are both a navigation task as well as a fighting task. Their importance maps are task dependent, for instance assigning lower importance to pictures on a wall when a player is engaged in opposing an enemy than when a player is navigating the virtual environment.

They found that the predictions of this importance map are of moderate quality during tasks where the player is less focused on a task, for instance when the player is navigating the environment. During periods when the player is very focused, i.e. during fighting against attacking enemies, up to 80% of the fixation time is deployed to enemies and explosions. This experiment reveals that under such conditions, objects representing enemies and explosions attract an exceptional amount of attention.

Since the game is a first person shooter, it can be expected that there is a strong bias toward fixating the center of the screen. This tendency can be exploited by encoding a measure of the degree of eccentricity from the spatial location of the objects. Bernhard's experiments show that this measure outperforms importance maps which rely on semantics, particularly in periods when there is less action in the game (e.g., during pure navigation tasks).

However, the highest predictive power is obtained when semantic and spatial information is combined. To keep the degrees of freedom low, object categories can, therefore, be clustered according to their importance values and then combined with eccentricity as an additional property.

25.9 Limitations

The work described in this chapter has opened a new avenue of gaze analysis methodology. Due to the fact that the authors were doing the very first steps into this field, many research efforts and technical investigations are required to bring their ideas forward to generally applicable tools.

One important technical limitation is the limited accuracy of the gaze-to-object mapping methods, which is discussed in Sect. 25.6.1.5. Another serious technical

problem is the collection and analysis of fixation statistics under the circumstance that each user has a unique game experience. As discussed in Sect. 25.7.1, there are no optimal strategies to fully compensate the latent effects caused by strong variations of the set of visible objects in the stimuli from one frame to another.

Moreover, the overall approach assumes a simplified world for games, which is composed of a set of objects with a clear semantic category and a clear geometrical outline. Though for many game objects this assumption holds true, commercial games frequently comprise more difficult content, e.g., large environment models or vegetations. Novel solutions need to be investigated, which allow decomposing all elements of a scene adequately, so that gaze can be analyzed according to the key features of major impact. Particular extensions to be considered are hierarchical decompositions of difficult objects, like trees or houses, and screen-space approaches to subdivide large models or background into regions in such a manner that features with a different response on visual attention can be spatially separated.

Finally, it is also important to further investigate the practical value of these tools. This includes evaluating their performance and value for a variety of different game types and examining in particular how these tools can be used to improve games.

25.10 Conclusion and Outlook

As this book discusses, evaluation of computer games is becoming increasingly important. While many chapters of the book focus on telemetry and game log data analysis, this chapter investigates eye tracking, which can be integrated with telemetry analysis and be used as one of the many tools at the disposal of the game user researcher. In addition to eye tracking and telemetry, there are also other techniques for gathering information from the player and to evaluate the gameplay experience, as discussed in several chapters in this book. Nacke et al. (2009) evaluated the experience of gaze-based interaction for example using different types of questionnaires; Chapter 24 in this book also reviews the use of questionnaires more extensively. More recently, eye tracking has been used in conjunction with psychophysiological data and game telemetry to evaluate the player experience. One way of evaluating the player experience is to gather quantitative data including biometric information from an electroencephalography (EEG), electromyography (EMG), galvanic skin response (GSR), heart rate (EKG), blood volume pulse (BVP), and breathing (Zammito et al. 2010). This is also a subject discussed by Nacke et al. and McAllister et al. in Chaps. 26 and 27 of this book. Using additional input techniques in addition to gaze could give even further information regarding the state of the player.

The main focus of this chapter, however, is on the mapping of fixation points obtained with an eye tracker to semantic objects as defined by game designers. The methods employed are necessarily more involved than recording screen shots, but

the opportunities for understanding game players' behavior are numerous. The work presented here only begins to scratch the surface. We see this approach as a viable technique for game designers to test their designs prior to bringing their products to market. At the same time, our enhanced and extended mapping techniques could form the basis for further research, for instance in understanding driver behavior in driving simulators.

About the Authors

Veronica Sundstedt is an Assistant Professor at the Blekinge Institute of Technology in Sweden where she coordinates the Computer Graphics research subgroup. She was previously a lecturer in the GV2 (Graphics, Vision, and Visualisation) Group in the School of Computer Science and Statistics at Trinity College Dublin, Ireland. She worked as a Postdoctoral Research Associate in the Department of Computer Science at the University of Bristol and the University of Bath, UK. She holds a Ph.D. in Computer Graphics from the University of Bristol and an M.Sc. in Media Technology from the University of Linköping, Sweden. Her research interests are in computer graphics and perception, in particular perceptually-based rendering algorithms, experimental validation, novel interaction techniques, and eye tracking technology. She organized and co-chaired the first Novel Gaze-Controlled Applications (NGCA) conference in 2011 and co-chaired the ACM APGV conference in 2007 and the Eurographics Ireland workshop in 2009. She is also on the editorial board of the ACM Transactions on Applied Perception. Veronica is Program Co-Chair for the ACM Symposium on Applied Perception (formerly APGV) in 2012 and the lead author of the book: *Gazing at Games: An Introduction to Eye Tracking Control*.

Efstathios Stavrakis is currently Visiting Lecturer at with the University of Cyprus. He holds a Ph.D. in Computer Science from the Vienna University of Technology (Austria) and has studied for an M.Sc. in Computer-Aided Graphical Technology Application and a BA (Hons) in Creative Visualisation at the University of Teesside (UK). He has conducted and published research in computer games, graphics and vision, eye-tracking and psychophysics, non-photorealistic rendering, as well as 3D audio rendering for VEs. He brings a wealth of experience in graphical algorithms, interface design and software development. Previously, he has held posts at the Technical University of Vienna (Austria), at INRIA Sophia Antipolis – Méditerranée (France) and the Glasgow School of Art (UK).

Matthias Bernhard is a Ph.D. Student at the institute of Computer Graphics and Algorithms of the Vienna University of Technology. He received a Bachelor degree Information Engineering at the University of Konstanz in 2004 and his Master Degree in Medical Computer Science at Vienna University of Technology in 2006. He follows an interdisciplinary perspective and his current research interests include bimodal perception and the role of visual attention in virtual environments.

Michael Wimmer is an Associate Professor at the Institute of Computer Graphics and Algorithms of the Vienna University of Technology, where he received an M.Sc. in 1997 and a Ph.D. in 2001. His current research interests are real-time rendering, computer games, real-time visualization of urban environments, point-based rendering and procedural modeling. He has coauthored many papers in these fields, and was papers co-chair of EGSR 2008 and Pacific Graphics 2012. He also co-authored the book “Real-time Shadows”.

Erik Reinhard received his Ph.D. in Computer Science from the University of Bristol in 2000, having worked on his Ph.D. at Delft University of Technology, as well as in Bristol. After holding a post-doctoral position at the University of Utah (2000–2002) and an Assistant Professorship at the University of Central Florida (2002–2005), he returned to Bristol as a lecturer in January 2006 to become senior lecturer in 2007. Erik founded the prestigious ACM Transactions on Applied Perception, and has been Editor-in-Chief since its inception in 2003, until early 2009. He is currently Associate Editor for this journal, as well as for Computers and Graphics. Erik is lead author of two books: ‘High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting’ and ‘Color Imaging: Fundamentals and Applications’. He is keynote speaker for Eurographics 2010, the Computational Color Imaging Workshop 2011 as well as the 6th European Conference on Color in Graphics, Imaging, and Vision (CGIV 2012). He is program co-chair for the Eurographics Symposium on Rendering 2011 and area co-chair for the high dynamic range imaging track at Eurographics 2011. Finally, his interests are in the application of knowledge from perception and neuroscience to help solve problems in graphics and related fields.

References

- Baylis, G. C., & Driver, J. (1993). Visual attention and objects: evidence for hierarchical coding of location. *Journal of Experimental Psychology: Human Perception and Performance*, 19(3), 451–470.
- Behrmann, M., Zemel, R. S., & Mozer, M. C. (1998). Object-based attention and occlusion evidence from normal participants and a computational model. *Journal of Experimental Psychology: Human Perception and Performance*, 24, 1011–1036.
- Bernhard, M., Stavrakis, E., & Wimmer, M. (2010). An empirical pipeline to derive gaze prediction heuristics for 3D action games. *ACM Transactions on Applied Perception (TAP)*, 8(1), 4:1–4:30.
- Bernhard, M., Zhang, L., & Wimmer, M. (2011). Manipulating attention in computer games. *IVMSP workshop, 2011 IEEE 10th* (pp. 153–158), Ithaca, NY.
- Canosa, R. L., Pelz, J. B., Mennie, N. R., & Peak, J. (2003). High-level aspects of oculomotor control during viewing of natural-task images. In B. E. Rogowitz & T. N. Pappas (Eds.), *Human vision and electronic imaging VIII. Proceedings of the SPIE in presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) conference* (pp. 240–251). Santa Clara, CA.
- Castiello, U., & Umiltà, C. (1990). Size of the attentional focus and efficiency of processing. *Acta Psychologica*, 73(3), 195–209.
- Cater, K., Chalmers, A., & Ledda, P. (2002). Selective quality rendering by exploiting human inattentional blindness: Looking but not seeing. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 17–24). Hong Kong, China.

- Cater, K., Chalmers, A., & Ward, G. (2003). Detail to attention: Exploiting visual tasks for selective rendering. In *Proceedings of the 14th Eurographics workshop on Rendering in EGRW '03* (pp. 270–280). Aire-la-Ville, Switzerland: Eurographics Association.
- Chaney, I. M., Lin, K.-H., & Chaney, J. (2004). The effect of billboards within the gaming environment. *Journal of Interactive Advertising*, 5(1), 37–45.
- Cunningham, D., & Wallraven, C. (2011). *Experimental design: From user studies to psychophysics*. Natick: A K Peters.
- De Graef, P., Christiaens, D., & d'Ydewalle, G. (1990). Perceptual effects of scene context on object identification. *Psychological Research*, 52(4), 317–329.
- Duchowski, A. T. (2003). *Eye tracking methodology: Theory and practice*. New York: Springer.
- Duncan, J. (1984). Selective attention and the organization of visual information. *Journal of Experimental Psychology. General*, 113(4), 501–517.
- Elazary, L., & Itti, L. (2008). Interesting objects are visually salient. *Journal of Vision*, 8(3:3), 1–15.
- El-Nasr, M. S., & Yan, S. (2006). Visual attention in 3D video games. In *ACE 06: Proceedings of the 2006 ACM SIGCHI international conference on advances in computer entertainment technology* (p. 22). New York: ACM.
- Eriksen, C., & St James, J. (1986). Visual attention within and around the field of focal attention: A zoom lens model. *Attention, Perception, & Psychophysics*, 40, 225–240.
- Haber, J., Myszkowski, K., Yamauchi, H., & Seidel, H.-P. (2001). Perceptually guided corrective splatting. *Computer Graphics Forum*, 20(3), 142–152.
- Hansen, D. W., & Ji, Q. (2010). In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), 478–500.
- Hayhoe, M. M., Shrivastava, A., Mruczek, R., & Pelz, J. B. (2003). Visual memory and motor planning in a natural task. *Journal of Vision*, 3(1), 49–63.
- Henderson, J. (2003). Human gaze control during real-world scene perception. *Trends in Cognitive Sciences*, 7(11), 498–504.
- Henderson, J. M., Weeks, P. A., & Hollingworth, A. (1999). The effects of semantic consistency on eye movements during complex scene viewing. *Journal of Experimental Psychology Human Perception & Performance*, 25, 210–228.
- Hillaire, S., Lécuyer, A., Cozot, R., & Casiez, G. (2008). Using an eye-tracking system to improve camera motions and depth-of-field Blur Effects in Virtual Environments. *VR* (pp. 47–50).
- Hillaire, S., Breton, G., Ouarti, N., Cozot, R., & Lécuyer, A. (2010). Using a visual attention model to improve gaze tracking systems in interactive 3D applications. *Computer Graphics Forum*, 29(6), 1830–1841.
- Hornof, A., Cavender, A., & Hoselton, R. (2003). Eyedraw: A system for drawing pictures with eye movements. *SIGACCESS Accessibility Computers* (pp. 86–93), Atlanta, GA, USA.
- Isokoski, P., Joos, M., Spakov, O., & Martin, B. (2009). Gaze controlled games. *Universal Access in the Information Society*, 8, 323–337.
- Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(11), 1254–1259.
- Itti, L., Dhavale, N., & Pighin, F. (2006). Photorealistic attention-based Gaze Animation. In *Proceedings of the IEEE international conference on multimedia and expo* (pp. 521–524). Toronto, Ontario, Canada
- Jacob, R. J. K., & Karn, K. S. (2003). Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In J. Hyönä, R. Radach, & H. Deubel (Eds.), *The mind's eye: Cognitive and applied aspects of eye movement research* (pp. 573–605). Amsterdam: Elsevier.
- James, W., & Anonymous. (1890). *The principles of psychology*, Vol. 1, volume reprint edition. New York: Dover Publications.
- Jie, L., & Clark, J. J. (2007). Game design guided by visual attention. In L. Ma, M. Rauterberg, & R. Nakatsu (Eds.), *Entertainment computing, ICEC 2007 in Lecture Notes in Computer Science* (pp. 345–355), Shanghai: Springer.

- Kenny, A., Koesling, H., Delaney, D., McLoone, S., & Ward, T. (2005). A Preliminary investigation into eye gaze data in a first person shooter game. In *19th European Conference on Modelling and Simulation*, Riga.
- Koch, C., & Ullman, S. (1985). Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 4, 219–227.
- Komogortsev, O., & Khan, J. (2006). Perceptual attention focus prediction for multiple viewers in case of multimedia perceptual compression with feedback delay. In *ETRA '06: Proceedings of the 2006 symposium on eye tracking research & applications* (pp. 101–108). New York: ACM.
- LaBerge, D. (1983). Spatial extent of attention to letters and words. *Journal of Experimental Psychology: Human Perception and Performance*, 9(3), 371–379.
- Land, M., Mennie, N., & Rusted, J. (1999). The roles of vision and eye movements in the control of activities of daily living. *Perception*, 28(11), 1311–1328.
- Lang, M., Hornung, A., Wang, O., Poulakos, S., Smolic, A., & Gross, M. (2010, July). Nonlinear disparity mapping for stereoscopic 3D. *ACM Transaction on Graphics*, 29(4), 1–75. doi:<http://doi.acm.org/10.1145/1778765.1778812>, URL: <http://doi.acm.org/10.1145/1778765.1778812>. New York: ACM.
- Luebke, D., Hallen, B., Newfield, D., & Watson, B. (2000). Perceptually driven simplification using gaze-directed rendering.
- Marmitt, G., & Duchowski, A. T. (2002). Modeling visual attention in VR: Measuring the accuracy of predicted scanpaths. In *Eurographics 2002, Short Presentations* (pp. 217–226). Saarbrücken, Germany.
- McDonnell, R., Larkin, M., Hernández, B., Rudomin, I., & O'Sullivan, C. (2009). Eye-catching crowds: saliency based selective variation. *ACM Transactions on Graphics*, 28, 55:1–55:10.
- Murphy, H., & Duchowski, A. T. (2001). Gaze-contingent level of detail rendering. In *Proceedings of EuroGraphics 2001 (Short Papers)*. EuroGraphics Association. Manchester, England.
- Nacke, L., Lindley, C., & Stellmach, S. (2008). Log who's playing: Psychophysiological game analysis made easy through event logging. In P. Markopoulos, B. de Ruyter, W. IJsselstein, & D. Rowland (Eds.), *Fun and games in lecture notes in computer science* (pp. 150–157). Berlin/Heidelberg: Springer. [10.1007/978-3-540-88322-715](https://doi.org/10.1007/978-3-540-88322-715).
- Nacke, L., Stellmach, S., Sasse, D., & Lindley C. A. (2009). Gameplay experience in a gaze interaction game. In A. Villanueva, J. P. Hansen, & B. K. Ersbøll (Eds.) *Proceedings of the 5th conference on communication by Gaze Interaction & COGAIN 2009: Gaze Interaction for Those Who Want It Most* (pp. 49–54), Lyngby, Denmark. The COGAIN Association.
- Nacke, L. E., Stellmach, S., Sasse, D., Niesenhaus, J., & Dachselt, R. (2011). LAIF: A logging and interaction framework for gaze-based interfaces in virtual entertainment environments. *Entertainment Computing*, 2(4), 265–273. <cc:title>Special Section: International Conference on Entertainment Computing and Special Section: Entertainment Interfaces</cc:title>.
- Navalpakkam, V., & Itti, L. (2005). Modeling the influence of task on attention. *Vision Research*, 45(2), 205–231.
- O'Sullivan, C. (2005). Collisions and attention. *ACM Transactions on Applied Perception*, 2(3), 309–321.
- Oliva, A., Torralba, A., Castelano M. S., & Henderson, J. M. (2003). Top-down control of visual attention in object detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP '03)*. Barcelona, Catalonia, Spain.
- Palmer, S. E. (1999). *Vision science: Photons to phenomenology*. Boston: MIT Press.
- Pelz, J. B., & Canosa, R. (2001). Oculomotor behavior and perceptual strategies in complex tasks. *Vision Research*, 41, 3587–3596.
- Peters, R. J., & Itti, L. (2008). Applying computational tools to predict gaze direction in interactive visual environments. *ACM Transactions on Applied Perception*, 5(2), 1–19.
- Poole, A., & Ball, L. J. (2005). Eye tracking in human-computer interaction and usability research: Current status and future prospects. In C. Ghaoui (Ed.), *Encyclopedia of human-computer interaction*. Pennsylvania: Idea Group, Inc.

- Rahardja, S., Farbiz, F., Manders, C., Zhiyong, H., Ling, J. N. S., Khan, I. R., Ping, O. E., & Peng, S. (2009). Eye HDR: Gaze-adaptive system for displaying high-dynamic-range images. *ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation in SIGGRAPH ASIA '09* (pp. 68–68). New York: ACM.
- Ramloll, R., Trepagnier, C., Sebrechts, M., & Beedasy, J. (2004). Gaze data visualization tools: opportunities and challenges. In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on* (pp. 173–180). London, UK
- Rothkopf, C. A., & Pelz, J. B. (2004). Head movement estimation for wearable eye tracker. In *Proceedings of the 2004 symposium on eye tracking research & applications in ETRA '04* (pp. 123–130). New York: ACM.
- Rothkopf, C. A., Ballard, D. H., & Hayhoe, M. M. (2007). Task and context determine where you look. *Journal of Vision*, 7(14), 1–20.
- Saito, T., & Takahashi, T. (1990). Comprehensible rendering of 3-D shapes. *SIGGRAPH Computation Graphics*, 24(4), 197–206.
- Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 symposium on eye tracking research & applications in ETRA '00* (pp. 71–78). New York: ACM.
- Sasse D. (2008). *A framework for psychophysiological data acquisition in digital games*. Master's thesis, Otto-von-Guericke-University Magdeburg, Magdeburg.
- Sennersten, C. (2004). *Eye movements in an action game tutorial*. Master's thesis, Lund University, Lund.
- Sennersten, C., & Lindley, C. (2008). Evaluation of real-time eye gaze logging by a 3D game engine. In *12th IMEKO TC1 & TC7 joint symposium on man science and measurement* (pp. 161–168). Annecy, France.
- Sennersten, C., & Lindley, C. (2009). An investigation of visual attention in FPS computer gameplay. In *Conference in games and virtual worlds for serious applications, VS-GAMES '09* (pp. 68–75). Coventry, UK.
- Simons, D. J., & Chabris, C. F. (1999). Gorillas in our midst: Sustained inattention blindness for dynamic events. *Perception*, 28, 1059–1074.
- Snowden, R., Thompson, P., & Troscianko, T. (2006). *Basic vision: An introduction to visual perception*. Oxford University Press, USA.
- Starker, I., & Bolt, R. A. (1990). A gaze-responsive self-disclosing display. In *CHI '90: Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 3–10). New York: ACM.
- Stellmach, S. (2007). A psychophysiological logging system for a digital game modification. Unpublished Internship Report, *Department of Simulation and Graphics*. Otto-von-Guericke-University, Magdeburg.
- Stellmach S. (2009). *Visual analysis of Gaze Data in virtual environments*. Master's thesis, Otto-von-Guericke-University Magdeburg, Magdeburg.
- Stellmach, S., Nacke, L., & Dachselt, R. (2010a). Advanced gaze visualizations for three-dimensional virtual environments. In *Proceedings of the 2010 symposium on eye-tracking research & Applications in ETRA '10* (pp. 109–112). New York: ACM.
- Stellmach, S., Nacke, L., & Dachselt, R. (2010b). 3D attentional maps: Aggregated gaze visualizations in three-dimensional virtual environments. In *Proceedings of the international conference on advanced visual interfaces in AVI '10* (pp. 345–348). New York: ACM.
- Stellmach, S., Nacke, L. E., Dachselt R., & Lindley C. A. (2010c). Trends and techniques in visual gaze analysis. *CoRR*, abs/1004.0258.
- Sundstedt, V. (2007). *Rendering and validation of high-fidelity graphics using region-of-interest*. PhD thesis, University of Bristol, Bristol.
- Sundstedt, V. (2010). Gazing at games: Using eye tracking to control virtual characters. *ACM SIGGRAPH 2010 Courses in SIGGRAPH '10* (pp. 5:1–5:160). New York: ACM.
- Sundstedt, V., Gutierrez, D., Anson, O., Banterle, F., & Chalmers, A. (2007). Perceptual rendering of participating media. *ACM Transaction on Applied Perception*, 4(3), 15.
- Sundstedt, V., Stavrakis, E., Wimmer, M., & Reinhard, E. (2008). A psychophysical study of fixation behavior in a computer game. In *APGV '08: Proceedings of the 5th symposium on applied perception in graphics and visualization* (pp. 43–50). New York: ACM.

- Sundstedt, V., Whitton, M., & Bloj, M. (2009). The whys, how tos, and pitfalls of user studies. *ACM SIGGRAPH 2009 Courses in SIGGRAPH '09* (pp. 25:1–25:205). New York: ACM.
- Tobii. (2006). *User manual: Tobii eye tracker; ClearView analysis software*.
- Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, 12(1), 97–136.
- van Zoest, W., & Donk, M. (2004). Bottom-up and top-down control in visual search. *Perception*, 33, 927–937.
- Wolfe, J. M. (1994). Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin and Review*, 1(2), 202–238.
- Wolfe, J. (2000). Visual attention. In K. K. De Valois (Ed.), *Seeing* (pp. 335–386). San Diego: Academic.
- Wolfe, J. M. (2007). Guided Search 4.0: Current Progress with a model of visual search. In Gray, W. (Ed.), *Integrated models of cognitive systems* (pp. 99–119). New York: Oxford University Press.
- Wooding, D. S. (2002). Fixation maps: Quantifying eye-movement traces. *Proceedings of the 2002 symposium on eye tracking research & applications in ETRA '02* (pp. 31–36). New York: ACM.
- Yarbus, A. L. (1967). Eye movements during perception of complex objects. In *Eye movements and vision* (pp. 171–196). New York: Plenum Press.
- Yee, H., Pattanaik, S., & Greenberg, D. P. (2001). Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transaction on Graphics*, 20(1), 39–65.
- Zammito, V., Seif El-Nasr, M., & Newton, P. (2010). Exploring quantitative methods for evaluating sports games. In *CHI 2010 workshop on brain, body and bytes: Psychophysiological user interaction*.

Chapter 26

An Introduction to Physiological Player Metrics for Evaluating Games

Lennart E. Nacke

Take Away Points:

1. Provides a brief introduction to physiological game evaluation.
2. Discusses the benefits and limitations of physiological measures for game evaluation.

26.1 Introduction

Do you remember insult swordfighting in Monkey Island? The moment when you got off the elevator in the fourth mission of Call of Duty: Modern Warfare 2? Your romantic love affair with Leliana or Alistair in Dragon Age? Dancing as Madison for Paco in his nightclub in Heavy Rain? Climbing and fighting Cronos in God of War 3? Some of the most memorable moments from successful video games, have a strong emotional impact on us. It is only natural that game designers and user researchers are seeking methods to better understand the positive and negative emotions that we feel when we are playing games.

While game metrics provide excellent methods and techniques to infer behavior from the interaction of the player in the virtual game world, they cannot infer or *see* emotional signals of a player. Emotional signals are observable changes in the state of the human player, such as facial expressions, body posture, or physiological changes in the player's body. The human eye can observe facial expression, gestures or human sounds that could tell us how a player is feeling, but covert physiological changes are only revealed to us when using sensor equipment, such as

L.E. Nacke (✉)

HCI and Game Science Group, Faculty of Business and Information Technology,
University of Ontario Institute of Technology, 2000 Simcoe Street North,
Oshawa L1H 7K4, ON, Canada
e-mail: lennart.nacke@acm.org



Fig. 26.1 An overview of game user research methods grouped together by similarity in a quantitative or qualitative and objective or subjective focus based on Mandryk (2008)

electroencephalographs (EEG), electromyographs (EMG), or galvanic skin response (GSR) recording systems. These player-focused body-related responses or physiological metrics are at the heart of this chapter.

26.1.1 Limitations of this Chapter

This book chapter was written with two audience types in mind: user researchers and graduate students. My goal is to give a brief overview of the field of physiological emotion research in games (more interesting as pointers for graduate students) as well as some how-tos for physiological recording (probably more useful for game user researchers). Keep in mind that this chapter cannot cover everything that you need to know about the background of recording physiological signals (Stern et al. (2001) is a better resource for this purpose) or give you all the information you will need to run physiological tests as a games user researcher. It is by nature a primer, something that hopefully gets you interested enough in physiological game research to start asking the right questions and look for the latest results in this growing field.

Figure 26.1 gives you an idea of the methods available for games user research and helps you locate at which part of the spectrum game metrics and physiological measures are (two of the more quantitative approaches available to game evaluators).

All of the methods mentioned in the figure are possible options for evaluating the player experience in a game depending on where your focus lies. Game metrics together with self-reported data from questionnaires or interviews can provide an additional cross-reference to physiological measures, which are still largely lacking validations for their use in games. But before we get there, let us review the emotional and physiological fundamentals for this type of games user research.

26.2 What Are Emotions?

Rosalind Picard mentions two types of signals in her “Affective Computing” book that need to be differentiated for emotion-recording systems: (1) expressive signals directly originating from a person and (2) non-expressive signals from environment and context of a person (Picard 1997). A physiological recording, for example, will not necessarily be able to differentiate between these types of signals. This is a problem that is similar to “situational stereotypy” (Lacey 1959), which is the idea that physiological responses depend on the experimental context. So, from a psychological view, the context in which players experience their emotions is as important as the game-related cues that trigger their emotions (for a player experience model that incorporates the idea of context see Engl and Nacke 2013). Picard (1997) notes that our expectations will influence our emotional perception, meaning that our body responses are shaped by our mental ideas and vice versa. A player’s own mood and emotions will influence their perceptions and cognitive processes (for an excellent review of how affective computing relates to psychological emotion literature, I recommend reading Calvo and D’Mello (2010)). At this point, the boundaries between user experience research and emotion research start to blur (Brave and Nass 2002). But let us keep the focus on emotions in the psychological sense for this introduction. So, where do we start, when we want to understand and distinguish emotions?

Emotion research is a huge field with journals such as *Emotion*, *Emotion Review*, *Cognition and Emotion*, or *IEEE Transactions on Affective Computing* at its heart, and the scope of this article forbids going into real depth here, but I want to give you some pointers about what the different views of emotions are. A general starting point for the interested emotion researcher are the following introductory articles: Kleinginna and Kleinginna (1981), Panksepp (2004), Bradley and Lang (2007), Russell (2003), Barrett (2006), Barrett et al. (2007), and Dalgleish et al. (2009). Of course, for those wanting to go in depth in this field, there are also several comprehensive handbooks available, on emotions (Lewis et al. 2010), on cognition and emotion (Dalgleish et al. 2000), on psychophysiological research (Cacioppo et al. 2007), on affective sciences (Davidson et al. 2003), on emotion and the affective sciences (Sander and Scherer 2009), and on emotion and mass media (Döveling et al. 2010), just to name a few. Finally, you should consult some comprehensive books on affective computing (Picard 1997; Scherer et al. 2010; Gokcay and Yildirim

2010; Pelachaud 2012) as well as affect and emotion (Panksepp 2004; Lane and Nadel 2002; Frijda 1986; Ekman and Davidson 1994) if you really want to get more familiar with the topic. By its nature, my overview is only brief and scratches the surface of more than a decade of emotion research.

There is no definitive taxonomy for emotions and there are many different ways of classifying emotions. One of the oldest theories of emotion is the James-Lange theory, which states that our emotion follows from experiencing physiological change first (James 1884; Lange 1912). According to this theory, when an outside event or object changes, it causes the physiological or visceral change, which then generates the emotional feeling. This theory has been challenged several times and continues to be criticized.

One of the first challengers was the Cannon-Bard theory, which offers an alternative sequence of emotion processing. After a feeling occurs, Cannon hypothesized that it triggers a behavior based on how the emotion is processed (Cannon 1927). The emotional perception influences the physiological reaction. How you think you feel will change your reaction to the feeling. This theory tries to account for a combination of high-level mental and low-level physiological responses when experiencing emotions.

Another emotional concept is the two-factor theory of emotions which is based on empirical observations (Schachter and Singer 1962). This theory considers mental processing to have a large influence on our individual interpretation of our body reactions to an event that caused them. According to Schachter, emotions stem from the interaction of two distinct factors: cognitive labeling and physiological arousal (Schachter 1964). Cognitive processes provide the framework in which individual feelings are processed and labeled, giving the state of physiological arousal positive or negative values according to the situation and past experiences. LeDoux (1998) provides an excellent overview of this in Chapter 3 of his book; specifically, he discusses some of these theories and the pathways of interpretation from the causing event to the resulting feeling.

For those more interested in modern theories of emotion that take into account that emotional processes can happen without the resulting emotional experience, I recommend Damasio (1994). It is also worth considering the multicomponent process emotion model (Scherer 1984), which constitutes that an emotion processing system consists of the five distinct subsystems: information processing, support, executive, action, and monitoring. This model is rooted in appraisal theory (Lazarus 1968), which denotes that emotional arousal from a stimulating event is ingrained in the meaning it has for the person perceiving it. Most studies regarding appraisal theory models of emotion have used verbal reports. This requires participants to engage in complex recall and imagination processes before they put their feelings into words.

To summarize, at the heart of most emotion theories are two basic concepts: (1) discrete emotional states and (2) dimensional (often biphasic) theories. Discrete emotional states date back to early ideas of the French philosopher René Descartes, who described basic emotions, such as joy, wonder, love, desire, hate and sadness. Later, Ekman (1972) would describe the appearance of the face for six distinct

emotions: surprise, fear, anger, disgust, sadness and happiness. A list that he extended later (Ekman 1992a, b). Other notable lists of discrete emotional states were contributed by Izard (1972) and Plutchik (1980). Plutchik also described a structural model of emotions (Plutchik 1991) has eight prototypic dimensions in horizontal (maximum intensity emotions at the top: ecstasy, acceptance, amazement, terror, grief, loathing, vigilance, rage) and vertical levels (lower located emotions are closer together and less intense).

While discrete emotions have a place in psychophysiology, they are often seen as broader concepts of underlying factors of a more affective nature, such as stimulus¹ appraisal (Scherer 1984), tendencies for action (Frijda 1986), or emotional expression through facial muscles (Ekman 1972). In psychophysiological research, dimensional models of emotion are more commonly used to conceptualize emotional facets. The most common model is the two-dimensional valence-arousal circumplex model (Russell 1980). The main criticism with this model is that the dimensions are not completely bipolar (Larsen et al. 2001), so alternative models were suggested, such as the positive activation and negative activation structure, that account for approach and withdrawal behavior (Watson et al. 1999). In this vein, another theory of positivity (appetition) and negativity (aversion) is offered by Cacioppo et al. (1999). In contrast to these newer models, early discussions of emotions tended to be completely biphasic, distinguishing between good and bad, positive and negative, appetitive and aversive, or pleasant and unpleasant. Only recently are we beginning to understand the complexity of affective processes that are often a blend of positive and negative feelings.

26.2.1 How Game Metrics Relate to Psychophysiological Emotion Induction

Emotions in a psychophysiological context can be understood as connected physiological and psychological affective processes, which can be induced by perception, imagination, anticipation, or action triggers (see Fig. 26.2). Perceptual emotions can be triggered by sensory information, such visual, acoustic, tactile, olfactory, or gustatory signals (Bradley and Lang 2007).

This distinction between emotional triggers is especially relevant when analyzing psychophysiological reactions together with game metrics. Only through the use of game logs that pinpoint exactly what game events were happening when, are we able to contextualize physiological reactions of players (see Nacke et al. (2008) and Kivikangas et al. (2011b) for descriptions of a logging systems that used player interaction logs together with physiological responses). Finding out what cues in the game induce a physiological reaction can be done by logging game metrics and sending game events coded as voltage triggers directly to physiological hardware

¹ A stimulus in psychological research is something (could be an event or an object) that evokes a body or mind response.

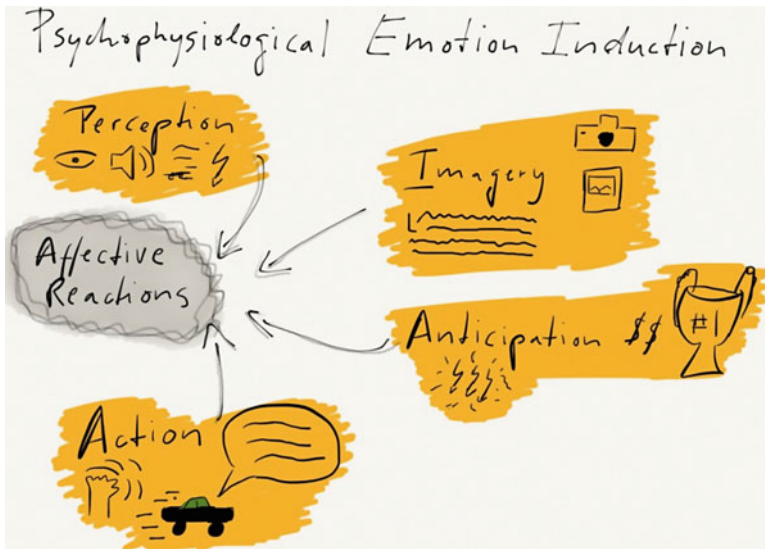


Fig. 26.2 Emotion inducers in the field of psychophysiology. Stimuli most often used in psychophysiological experiments come from these contexts

(Kivikangas et al. 2011b). Alternatively, one can triangulate player and event logs with physiological data as long as they contain a timestamp that is synchronized with the physiological timestamp (a procedure which can be difficult if several computers are involved; in this case networked time synchronization is suggested). Other approaches include a “manual” correlation of the physiological data with player events using video data. Here, several videos (usually of the player’s face, an in-game capture, an event log, and physiological graphs) are watched after a player session and events of interest are identified and scored manually in physiological data processing software. We will talk more about triangulation and data storage procedures for physiological data later in this manuscript. Before we talk about detailed physiological recording procedures, we need to understand the field of psychophysiology.

26.3 What Is Psychophysiology?

Psychophysiology is a research field where body signals, so called physiological responses, are measured to understand what mental processes are connected to those bodily responses (see Darrow 1964; Andreassi 2006; Hugdahl 1995; Cacioppo et al. 2007 for more definitions). I will refer to this as physiological metrics in this chapter. In this area of research, we are studying body signals to get an idea of what our mind was doing at that point. We are studying brain-behavior relationships that are guided by activity in the nervous systems (Hugdahl 1995). This makes psychophysiology a

useful tool for evaluating excitement, emotion, or mental workload in games by conducting experiments. However, one has to keep in mind that valid experimentation requires much caution and preparation. In physiological experimentation, we have to balance ecological validity of our experimental environment with possible distractions that need to be controlled for.

Most of our body responses are spontaneous. This means they are difficult to fake, which makes physiological measures more objective than, for example, behavioral gameplay metrics, where a participant is able to fake doing an activity while cognitively engaging in another. One could say they allow the least biased assessment of how a player is reacting to gameplay actions compared to other game user research methods. They are also recorded continuously, meaning they do not interrupt a player's gameplay session. Physiological metrics are vast amounts of data, which become meaningful only when analyzed using the correct context and correct signal processing procedures (Mandryk 2008; Nacke 2009). For example, as a game designer we want to create meaningful decisions that involve some tradeoff of game resources (e.g., resource trades, weighing risk against reward, and choosing an appropriate action) (Brathwaite and Schreiber 2008). Here, emotional decisions can make playing games more fun. In these game decision situations, physiological metrics give you an objective way to assess a player's emotional response. We can get an idea about the emotional state of a player based on physiological metrics and this helps us inform game designers. In case of our example, we would know whether the designers have succeeded in causing an emotional response in the player with the decision options that they provided. However, again you have to keep in mind that this type of quantitative data has to be interpreted to make correct design suggestions, which leaves room for interpretation bias of the game user researcher.

Without a high level of experimental control, physiological data is volatile, variable, and difficult to interpret. For example, if a think-aloud protocol is applied when recording physiological metrics, a researcher risks influencing heart rate and respiration. When interpreting physiological metrics, it is important to understand the relationship between what happens in your brain (the psychological effect or mental process) and what your body tells us (the physiological variables, such as EEG, EMG, EDA). Cacioppo et al. (2007) note that there are five general relations between mental processes and body responses that we need to understand. The following relationships are distinguished:

- **The one-to-one relationship.** One mental process is directly associated with one body response and vice versa. This type of relationship would allow you to identify a mental process based on a body response and it is rarely possible.
- **The one-to-many relationship.** One mental process is associated with many body responses. Here, we cannot make draw a conclusion regarding mental processes.
- **The many-to-one relationship.** Many mental processes are associated with the same body response. While this scenario is worse than a one-to-one relation, it is the one most often used in physiological evaluation. It allows us to make assumptions of mental processes based on a body response.

- **The many-to-many relationship.** Many mental processes are associated with many body responses. Again, this type of relation does not allow for a conclusion of mental processes based on body responses.
- **The null relationship.** There is no relationship or association between mental processes and body responses.

The most common case in physiological evaluation is the many-to-one relationship, where one body response may be associated with many mental effects or processes. Therefore, we must keep in mind that a direct mapping of a discrete emotional state is not possible (and it is debatable whether discrete emotional states even exist, although we will not touch on this discussion) and body responses must be understood as elements of sets with fuzzy boundaries. When we measure body signals, we are measuring essentially the operation and activity of muscles, nerve cells, and glands.

26.4 Physiological Response Metrics of Players

To understand how physiological measures work on the human body, we need to take a quick neurobiological look at how our bodily reactions are organized. On a macro level, bodily operations are controlled by our nervous system, which is split into two parts, the central nervous system (CNS) and the peripheral nervous system (PNS). The CNS consists of big brain (cerebrum), little brain (cerebellum), and spinal cord. It manages all the information received from the whole body and coordinates body activity accordingly. The CNS is well protected by the skull and spine bones, which also makes it difficult to access outside of the body. The PNS includes all nerve cells outside of the CNS. You could say that its main job is to connect the CNS to the rest of our body. To use an example from musical theater, you can imagine the CNS having the same functions as the conductor in a concert, whereas the PNS would be the orchestra.

Since most of our physical sensations are transmitted through the PNS, we are able to measure its reactions on our skin. The skin is the place where most physiological sensors are applied. More on the micro level, the PNS is split into the somatic and the autonomic nervous system. It is enough to say here that the somatic nervous system regulates body activity that we have under conscious control such as deliberate movement directly through our muscles. The autonomic nervous system (ANS) is more exciting for physiological evaluation because it takes main care of our unconscious, visceral responses. These responses are hard to get with classic game user research methods and physiological metrics can really shine here. In the ANS, just like in a good game, we have two opposing players, the sympathetic nervous system and the parasympathetic nervous system. The former is our emergency response system that triggers fight or flight reactions while the latter manages our relaxation, resting, and digesting. It is important to keep those two players in mind, when we look at how we measure emotion with physiological sensors.

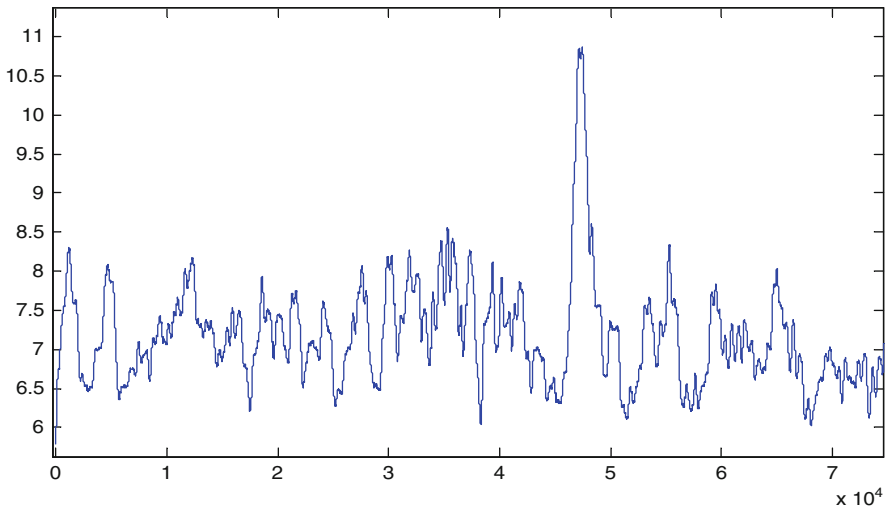


Fig. 26.3 Raw psychophysiological signal with a baseline offset (e.g., EMG)

The PNS is particularly useful for measuring stimulation, but not so much when it comes to measuring emotion itself. However, by detecting slightest muscular movements in the face with physiological sensors, we are able to assess emotion based on facial expression. For example, a frowning face would express negative emotion, whereas a smiling face would express positive emotion. Both reactions will show as spikes in the data of the physiological sensor that is applied to the corresponding region of the face. As game user researchers, we are also interested in feelings and the experience players are having when interacting with a game. Therefore, we cannot solely rely on physiological metrics for player testing, but we have to accompany them with questionnaires or other contextual recording techniques (e.g., interviews, video observation, game metrics) to get a better idea of player experience. However, the basic tenet of physiological experimentation still holds true: we measure the physiological response (in addition to other subject responses) while manipulating a behavioural factor, often an element of gameplay. For an overview of recent game research studies that are investigating psychophysiology in games, see Kivikangas et al. (2011a).

26.4.1 Physiological Signal Processing Primer

Raw physiological signals, such as EEG and EMG (EDA is a bit different as we will note later), represent an assembly of positive and negative (i.e., an oscillating) electrical voltage. Important traits of a physiological signal are the frequency (number of oscillations) and their amplitude (maximum positive or negative voltage). An example of a raw psychophysiological signal can be seen in Fig. 26.3.

In a psychophysiological recording graph, we can see two dimensions. The abscissa shows the recording time (often ms) and the ordinate displays the amplitude (often in μV). Raw EMG signals, especially when the sensors are applied to larger muscle sites are characterised by activity bursts (when the muscle contracts) and baselines during the muscle resting periods. For facial EMG, the signal is not quite as distinguishable in experimental situations, since an alert participant is rarely completely relaxed and there is always some activity visible in the raw EMG (depending on the resolution and filters of the recording hardware). Recording baseline noise before exposure to an experimental stimulus and subtracting this from the signal is a common method used for removing unwanted noise from the recordings before starting to filter the signal. When first looking at physiological signals, do not be surprised by seeing many negative numbers in your first recording (negative values are plotted upwards as a neurophysiological convention). Signal filtering from the raw signal toward a normalized signal usually follows the following steps (Tassinari et al. 2000):

1. Rectify the raw signal using a technique called RMS or root mean square, which equals the quadratic mean of a number. Applying an RMS transformation to the raw signal folds makes it easier to view and understand.
2. Sometimes the physiological recording hardware (the so-called amplifier) already has a bandpass filter (10–500 Hz) built in. If not, then at least for EMG signals a lowpass filter of 500 Hz should be applied to remove noise from the amplifier hardware. The decision of whether to use a 10, 20 or 30 Hz high pass filter depends on how much the researcher wants to attenuate weak signals (30 Hz gets rid of cross-talk and other noise such AC power). The 10–500 Hz bandpass filter is sufficient for most EMG applications and can easily be implemented in MATLAB using a digital third order Butterworth bandstop filter. For EEG data, using a smaller range between 1 and 40 Hz would be advisable depending on whether gamma frequencies (30–50 Hz) are used in the analysis. Since EEG analysis works on lower frequencies, a notch filter can be applied as well to remove 60 Hz noise.²
3. Finally the signal is often smoothed using a moving window technique, where based on a time window defined by the researcher data is averaged within the moving window. This often called moving average or average rectified value.
4. For EEG, a next step would be to calculate average power estimates with a Fast Fourier Transformation (FFT). Since most EEG analysis is more complicated and warrants a chapter of its own, we will not go into depth here.

Depending on what kind of statistical analysis is later done with the physiological signal, it can be logarithmically (log or ln) transformed to eliminate skew in the data distribution. The rest of the analysis is done depending on the experimental setup and using statistical methods.

² 50/60 Hz is the electrical energy frequency that can come from lights, power supplies and other devices in your experiment environment.

When designing a games user research study that could involve physiological sensors, you have to pick wisely which sensors to use and whether to use sensors at all. Some game user research questions can be answered with other methods, depending on what you want to know about the user experience. Skin conductance level is correlated with psychological arousal (Prokasy and Raskin 1973), but so are cardiovascular measures, such as blood volume pulse, higher heart rate, and respiration. If you wanted to look at mental effort and task load, you could resort to a subjective measure like the task load index (Hart and Staveland 1988), look at multivariate EEG measures (Smith et al. 2001), decreased heart rate variability, or more dilated pupils (using eye tracking technology), brow or jaw muscle activity (Waterink and van Boxtel 1994). We will also later in this chapter talk about using facial EMG to assess positive or negative valence of emotions to indicate whether a game action is perceived as positive or negative. With all these measures available and each of them using different signal processing, the task of choosing the right one might seem daunting at first. For people getting started with physiological measures, I recommend sticking to the basic skin conductance and EMG measures presented in this chapter.

For some games sensors might be a better fit than others. In our experience, action games that produce a visceral experience are generally a good fit for physiological measures (Nacke 2009). It remains to be shown whether this is a useful tool for casual games as well, since recent reports have yet to make a strong argument for the method in this context (Gualeni et al. 2012). If you decide that physiological sensors are your method of choice for your games user research question, you should choose a sensor that will not alter the player experience through its application, but one that is sensible to the effects that you want to measure.

26.4.2 *Electroencephalography (EEG)*

There are many myths surrounding EEG as a measure of brainwave activity of the human body. Participants unfamiliar with this technique may assume that you are able to find out exactly what they are thinking or even get graphic representations of their thoughts. While recent research in the latter (i.e., reconstructing visuals from brain activity using magnetic resonance imaging [MRI]) has been impressive (Nishimoto et al. 2011), the reality of EEG measures is a little bit more abstract than one might think. Compared to other techniques of analysing the CNS response, such as functional MRI or positron emission tomography (PET) scans, EEG, can be considered less invasive and easier to apply. The advantage of EEG for brain activity measurement over these other techniques is its millisecond resolution, which allows studying physiological responses in real time. A slight disadvantage of EEG to the other approaches is its spatial resolution, which is constrained, for example, by a low signal-to-noise ratio and limited spatial sampling.

Example Experimental Protocol for Attaching EEG Electrodes

1. When selecting or inviting participants for physiological studies that use surface electrodes such as EEG or EMG, it is a good idea to **screen participants for hair growth**. EEG usually has to be applied directly on the scalp (head skin surface) of the participant, the more hair a participant has, the more electrode gel you are likely to use. When using male participants for facial EMG, a beard might also be problematic when trying to apply the electrode directly to the skin. Adhesion is reduced when much or thick hair is present, especially when recording under humid conditions or with skin types prone for sweating. Also, no chewing gum for participants.
2. Most dry electrodes do not require extensive **cleaning of the skin**, although it is recommended for hygienic reasons to clean the skin before and after attaching electrodes, but not with soap. A soft cleaning with an alcohol pad or conductive cleaning paste is usually sufficient for removing dead skin cells.
3. If not using pre-gelled electrodes, the **electrode needs to be gelled** (or wetted for some toy EEG devices) for optimal skin contact. EEG electrodes are often snapped into a cap that is worn (and needs to be correctly placed) on a participant's head to ensure correct alignment of all electrodes.
4. Depending on thickness of the electrode cables, having **surgical tape** on site is invaluable for making sure that the **cables** are closely **attached** to the participant and do not move around during recording.
5. After the recording or experiment is done, the **electrodes need to be removed** from the participant as soon as possible to minimize discomfort.
6. All **equipment** that was in contact with the participant needs to be **washed** (sensor cap and straps) or thrown away (disposable electrodes).

In EEG, electrodes are placed on a participant's head. Their location and alignment is standardized in the 10–20 system (Jasper 1958) (or the 10–10 EEG sensor placement system (Chatrian et al. 1988)). Often EEG systems ship with caps that take care of this alignment by having electrode inlets sewn into headgear that looks like a swimming cap. EEG measures slight electrical activity, such as the signals generated by neural activity in the brain. There is a wide range of different measurement devices available for this type of physiological measure, ranging from a more sophisticated medical grade headcap setup with large density electrode arrays (from 32 to 256 electrodes) and simpler devices that have less electrodes and therefore less spatial accuracy but similar time accuracy. Some really cheap EEG devices sell for lower than \$1,000 (e.g., Neurosky, Emotiv). Most of these devices compute affective and cognitive states such as attention, engagement, boredom, meditation, frustration, or long- and short-term excitement. Be aware that these computations are not openly available and they are mostly a black box for researchers.

EEG lets us record electrical activity on the head that relates to brain activity. We usually distinguish brain activity by using the amplitude and frequency of the signal in comparison to a reference location. Amplitude describes the size of the signal,

while frequency refers to the speed of signal cycles. EEG devices compute brain waves in different frequency bands, such as alpha (e.g. 8–13 Hz), beta (e.g. 13–30 Hz), theta (e.g. 4–8 Hz), delta (1–4 Hz), and sometimes gamma (30–50 Hz).³ Alpha activity is associated with relaxation and lack of active cognitive processes; it has also been tied to information and visual processing. Beta activity is related to alertness, attention, vigilance, and excitatory problem solving activities. Theta activity has been related to decreased alertness and lower information processing, however, frontal midline theta activity in the anterior cingulate cortex scalp area is linked to mental effort, attention, and stimulus processing. Delta is most prominent during sleep, relaxation or fatigue. Gamma activity is still largely unexplored. While these associations come from research in medicine and psychology, they make it easier to evaluate a game based on the EEG activity. For example, if you notice increased beta activity during gaming, it could be linked to player attention and increased arousal during a focused gaming task.

A major disadvantage of early EEG methods was the placement of the electrodes with gel. Many budget-type EEG devices got rid of the gel and have dry electrodes. This minimizes discomfort by providing a comfortable fit on the head.

EEG is difficult or impossible to measure when there is movement involved. The electrodes might move on the head while the player is moving. This leads to artifacts in the EEG data. Therefore, some games are not very suited for EEG evaluation (e.g., Guitar Hero, Kinect, or Wii games). Movement artifacts are a problem of all physiological measures, but are especially problematic with EEG as we are interpreting very low electromagnetic activity. It is important to apply proper filters to your EEG data, so that no interferences are recorded in the EEG signal (e.g., often a 50/60 Hz notch filter is used to exclude interference signals).

In addition, as with all physiological measures, EEG measures should be recorded with a baseline. For example, you could record this at the start of your session and let the player do nothing but stare at a cross on a grey background. This allows getting rid of the noise in your EEG signal. A final problem with EEG as a method is the difficult interpretation of the data. For example, when delta activity is increased in a playing session, do we argue that the game is relaxing or that it is boring and fatigue-inducing? It is quite important to keep one's game design goals in mind when doing this type of evaluation. Relating this data with other measures is paramount for a solid interpretation. Table 26.1 shows the pros and cons of EEG.

26.4.3 *Electromyography (EMG)*

An EMG measures whether our muscles are active or not. Therefore an EMG electrode attached to the surface above a muscle is able to sense even the slightest activation of this muscle (Bradley et al. 2001; Lang 1995). Whenever we flex a

³ Another way of analysing EEG is through Event-Related Potentials or Mu Rhythm, which I do not cover in this chapter.

Table 26.1 Pros and cons of measuring EEG

PRO	CON
Great time resolution	Low space resolution
Deep cognitive insights	Gel-based caps and conductivity
Quantitative data	Movement artifacts
Small system setup	Data needs proper filtering
Different analyses possible with the same data set	Difficult to interpret
	Expensive

muscle on our body, this produces a difference in electrical activity or isometric tension which is measurable by EMG. While EEG measures activation in the CNS, EMG is all about measuring PNS activation. Since most muscles can be directly controlled, EMG is a measure of high interest for interacting with computers in a more natural way (Nacke et al. 2011).

However, the most common use for evaluating games is through facial EMG (Fridlund and Cacioppo 1986), which measures the activation of specific facial muscles responsible for displaying our positive or negative reactions to an emotional moment in a game (Hazlett 2006). In particular, physiological game research has focused on using brow (corrugator supercilii) to indicate negative emotion and cheek muscle (zygomaticus major) to indicate positive emotion (Mandryk et al. 2006) or even fun and flow in a game (Nacke and Lindley 2008). For longer term evaluation (say over a few minutes of gameplay), the eye muscle (orbicularis oculi) has also proven helpful in registering high arousal pleasant emotions (Ravaja et al. 2008).

In game user research using facial EMG to assess emotions, we have recently found the threshold of the total signal average with added standard deviation (Hazlett 2008) helpful to identify significant positive and negative gameplay moments (see Fig. 26.4 for a visualization). Hazlett used this to calculate an EMG ratio for a game with the total time spend in brow muscle activation (negative) or cheek muscle activation (positive). In my research group,⁴ we have used this positive and negative gameplay time measure together with video observation and biometric storyboards (Mirza-Babaei et al. 2012) to identify key positive and negative moments during gameplay. Combined with gameplay logs, we can correlate this negative and positive gameplay response time with behavioral events, such as button presses, navigational interactions and gameplay actions. We are working on automating this scoring process and are working on validating this procedure and making the tools available for the game industry. While this will not allow the fine grained level of details that a mixed methods gameplay video analysis will provide, a gameplay metrics based scoring system that takes into account physiological responses will definitely be useful for the game industry.

Similar to EEG, EMG uses silver-silver chloride electrodes (see Fig. 26.5) because they have only a small measurement error, little drift potential, and minimal polarization. EMG electrodes are applied to the surface of the skin and will also

⁴<http://hcigames.businessandit.uoit.ca>

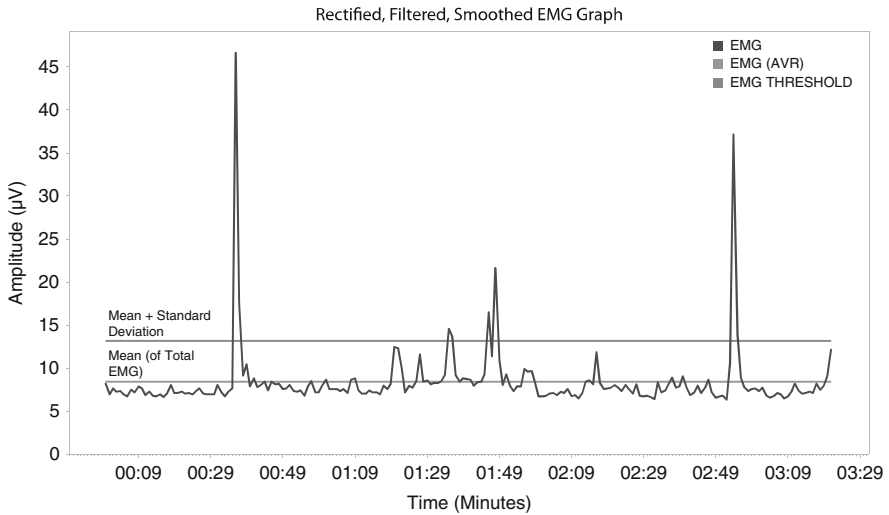


Fig. 26.4 Example of a filtered, smoothed and rectified EMG graph, showing thresholds for EMG analysis. Hazlett (2008) suggested using a threshold of average (i.e., Mean) EMG Amplitude plus Standard Deviation for finding positive measures

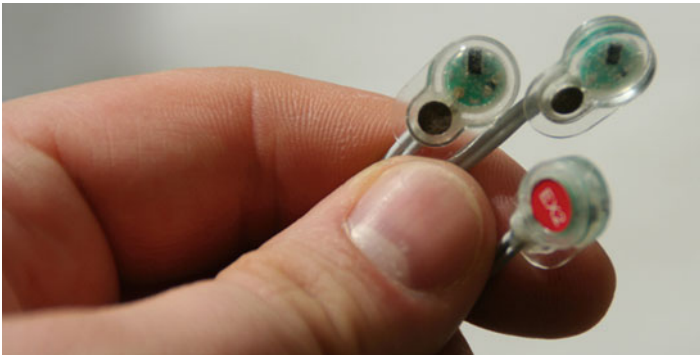


Fig. 26.5 An example of EMG electrodes (silver-silver chloride)

need a reference (if part of a larger system this reference can be on the head or close to the EMG electrodes). In measuring facial EMG one risk is to pick up muscle activity that is not related to the muscles that you would like to measure, such as cheek muscle for positive emotions or brow muscle for negative emotions.⁵ In clinical settings, EMG electrodes might be placed under the skin surface to eliminate muscle interference, but these are not appropriate in a game user research setting. However, screening for facial hair is recommended, since body hair can cause interference

⁵ For example, participants in an experiment cannot chew gum, laugh, or talk during facial EMG, because this will introduce large artifacts in your EMG data.

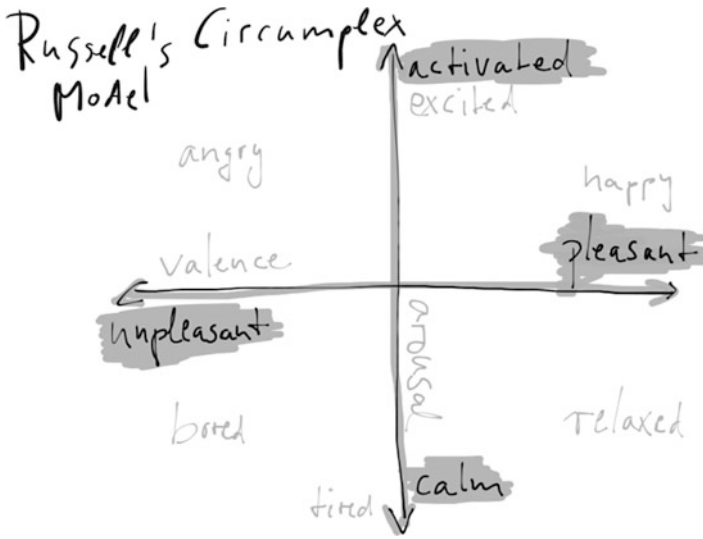


Fig. 26.6 Two emotion dimensions (valence and arousal) in the circumplex model from Russell (1980)

Table 26.2 Pros and cons of measuring EMG

PRO	CON
Great time resolution	Muscle and movement interference
Best way to measure emotion	Data needs proper filtering
Quantitative data	Electrode placement in the face
Easy signal analysis	Difficult to get a natural measurement
More precision than face cameras	Expensive

with EMG signals. Since muscular signals are amplified from microvolts, careful signal processing has to be done on EMG data before it is interpreted.⁶

Figure 26.6 shows how emotions are usually interpreted in psychophysiology on a two dimensional model (Russell 1980). We find that by measuring these face muscles we are able to get an idea of pleasant or unpleasant emotions along one axis of this model. This is called emotional valence assessment as we are able to show whether an emotion was evaluated by a player as pleasant or unpleasant.

While facial recognition software or direct observation would also allow the analysis of facial expressions and therefore the mapping on emotions, the software or the observer often miss less salient expressions, which are picked up by physiological measures. See Table 26.2 for pros and cons of EMG.

⁶ The usual processing procedure is signal smoothing (often at half of the recording frequency, for example 0.5 s at 2 kHz recordings), baseline subtraction, and sometimes a logarithmic normalization. Depending on the system, an additional bandpass filter (high: 10Hz, low: 400Hz) or a Butterworth lowpass filter of 500Hz are necessary.

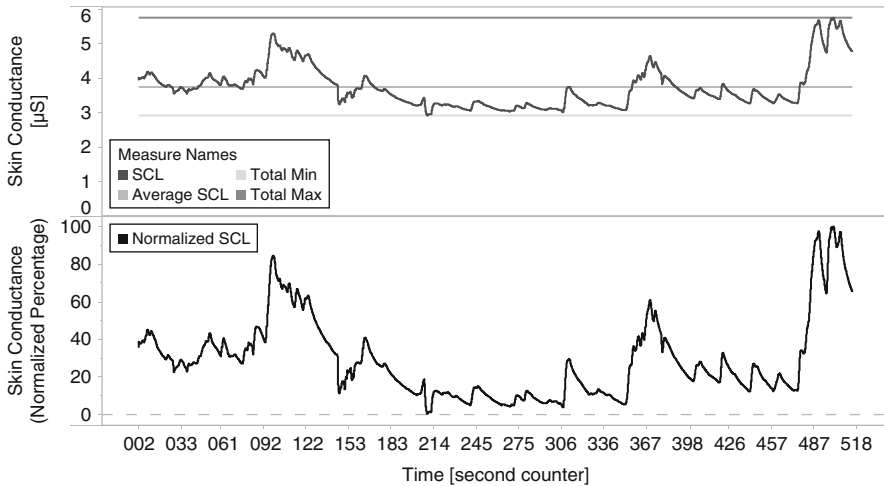


Fig. 26.7 A skin conductance level (SCL) graph for an example EDA recording. The *upper graph* shows the raw skin conductance level measured in μS sampled at 32 Hz over a total time of ~ 513 s (16,415 samples) together with the total average SCL as well as total minimum and total maximum. The *second graph* shows a normalized version of the skin conductance level based on the equation described below

The analysis of EMG signals is straightforward, as usually after application of some filters, we are already able to compare the signals. More activity on the cheek muscle relates to positive emotion, more activity on the brow muscle relates to negative emotion. However, EMG measures in the face of a player mean that there are electrodes attached to the player's face while playing, which make this measure intrusive although often the electrodes and cables can be easily taped to the player's head to remove discomfort and reduce movement artifacts. One thing to keep in mind is that just by feeling the electrode on their face, players might be feeling the need to elicit more pronounced muscle movements when playing. This might lead to unnatural signals, which could make data interpretation more difficult (if no video recording is available to check for this problem).

26.4.4 Electrodermal Activity (EDA)

EDA relates to how excited we are when we are exposed to a stimulus, such as playing a game. When measuring the skin conductance level (SCL) over time, we refer to this as measuring the EDA of the skin (see Fig. 26.7), but when measuring the direct response to an event, we call this galvanic skin response (Boucsein 1992). In any case, EDA measures changes in the passive electrical conductivity of the skin relating to increases or decreases in sweat gland activity. These fluctuations are caused by a person getting aroused by something that they see or do.

Most of us have seen EDA measures in movies branded as lie detector tests. EDA measures are attached to the fingers, palms or toes because the sweat glands in those body areas are more likely to react to changes in the PNS (sympathetic vs. parasympathetic activity). Since we are measuring the differences in conductivity, we only need two electrodes, which make EDA a very easy physiological measure to prepare and apply. EDA electrodes are prone to movement artifacts just like other psychophysiological measures, but because of their location (hand or feet) special care has to be taken in the preparation of steering controls of the game. If a regular game controller is used and the EDA sensor is applied on the palm of the hand, movement artifacts are likely to occur, so using the feet or fingers of the non-dominant hand would be a better location (or the side of the palm of the hand). EDA is also very easy to interpret, since it almost has a one-to-one relationship with physical arousal. However, individuals are different in their SCL, so a comparison between people is only possible with normalized data. SCL can be normalized by calculating each sample as a percentage of the entire span of EDA, using the min and max values over all samples for one participant (Mandryk 2008; Lykken and Venables 1971). The equation below shows how to normalize your SCL data at any point in time (SCL_{now}) as a percentage, given that you know the maximum (SCL_{max}) and minimum (SCL_{min}) value of your EDA data.

$$SCL_{normalized} = \frac{SCL_{now} - SCL_{min}}{SCL_{max} - SCL_{min}} \times 100$$

An equation of normalizing skin conductance level (SCL) based on Mandryk (2008).

Another benefit of this measure is the inexpensive hardware that usually comes at a fraction of the cost of a research-grade EEG setup. Many modern EDA systems use dry electrodes and some EDA setups for example allow quickly attaching the electrodes to the little and ring finger with a Velcro strap. This makes EDA are very handy measure for game user research.

Analyzing SCL can be done in a macro (EDA over larger chunks of playtime) or micro fashion (GSR related to events). When analyzing the response to a direct event, one needs to take into account that EDA is a relatively noisy signal that also has some delay in response to a stimulus (often around 5 s). After a galvanic skin response is registered, there is also a decay or recovery time during which no further event responses will be registered (or the responses are registered together). In addition to this, EDA tends to drift over time, possibly a result of the hands or feet getting sweatier. A good way to make sure this drift does not affect the EDA data too much, is to make sure to have resting periods between different gameplay sessions. While it is pretty clear that EDA indicates physical arousal, there is still some interpretation effort required as to what this stimulation comes from. Is it really from a game stimulus or are environmental factors contributing to the response? This is why planning and controlling physiological experiments is very important. Possibly confounding factors, such as high physical activity, loud noise, caffeinated substances, bright light, and things moving in the background should be avoided at all costs when running a physiological study. Table 26.3 shows the pros and cons of the EDA physiological metric.

Table 26.3 Pros and cons of measuring EDA

PRO	CON
Cheap hardware	Noisy signal
Easy to measure	Large individual variation
Easy to interpret	Baseline and response fluctuations
Less intrusive than other biosensors	Slow decay over time

Table 26.4 Pros and cons of cardiovascular measures

PRO	CON
Heart rate is easy to measure	Intrusive sensor
Heart rate hardware is cheap	Affected by many different things
Cardiovascular measures are established and prominent	Heart rate variability has a complex analysis procedure

26.4.5 *Cardiovascular Measures*

There are many cardiovascular measures available for physiological evaluation and all of them relate to the heart rhythm, its changes, and how this influences the physiological state of a human. The most common measures are electrocardiography (ECG), heart rate (HR), interbeat interval (IBI), heart rate variability (HRV), blood volume pulse (BVP), and blood pressure (BP). While physiological electrodes are necessary for all measures, blood pressure is not a real-time measure and also usually used in a medical context and has not been shown to be of relevance to game user research.

ECG measures the electrical activity caused by the heart pumping blood and is measured with three electrodes or leads, which are positive, negative and neutral and are usually applied to the upper body area. This can be considered a somewhat intrusive area for sensor placement depending on a participant's comfort level.

Heart rate is understood as the number of heart beats per time unit (usually measured in beats per minute). The amount of heart beats during a time unit is an interesting metric as is the time between the beats, the IBI. If IBI decreases, HR increases and this has been tied to increased information processing and emotional arousal. So, IBI and HR are two related measures. However, HR variability is a more complicated measure with a complex analysis procedure. In HRV, we are looking at differences in the IBI over time and analyze frequency changes. In general, we need to keep in mind that cardiovascular measures are intrusive to measure accurately and they are affected by many things, such as physical activity. Table 26.4 shows the pros and cons of physiological cardiovascular measures.

26.4.6 *Other Physiological Measures*

There are a number of other physiological measures not covered in this introductory chapter, such as respiratory sensors, eye trackers, temperature sensors, and brain imaging techniques. Another chapter in this book deals with eye tracking techniques

in depth. And there is good other introductory literature available for this field (Duchowski 2007) as well. In addition, for more details on respiratory sensors and cardiovascular measures, other sources are available (Mandryk 2008).

26.5 Case Study: Physiological Measures of Sonic Gameplay Experience

This case study explains an experiment I conducted together with colleagues during my Ph.D. studies and a part of which was published in the journal *Interacting with Computers* (Nacke et al. 2010). Our initial research question behind this study was whether we can investigate the effects of sound and music in games on physiological measures and subjective measures of player experience. The research was conducted using a modified version of the first person shooter *Half-Life 2* (Valve Corporation, Bellevue, WA, USA). The modified level was designed for a playing time of 10 min. The game mod was played four times in different sound and music conditions. A first-person shooter is an excellent environment to conduct this type of research, since gameplay is highly arousing and visceral, which we hoped to be likely to yield physiological responses.

26.5.1 Metrics Used in This Study

Facial electromyography (EMG) was used to record the activity from left orbicularis oculi (eye), corrugator supercillii (brow), and zygomaticus major (cheek) muscle regions using BioSemi flat-type active electrodes with sintered Ag-AgCl (silver/silver chloride) electrode pellets having a contact area 4 mm in diameter. The electrodes were filled with low impedance highly conductive Signa electrode gel (Parker Laboratories, Inc., Hellendoorn, The Netherlands). The raw EMG signal was recorded with an ActiveTwo AD-box at a sample rate of 2 kHz and using ActiView acquisition software, and afterwards filtered in BESA (MEGIS GmbH, München, Germany) using a low cutoff filter (30 Hz; Type: forward, Slope: 6 dB/oct) and a high cutoff filter (400 Hz; Type: zero phase, Slope: 48 dB/oct). Electrodermal activity (EDA) was measured using two passive Ag-AgCl (silver/silver chloride) Nihon Kohden electrodes (1 mA, 512 Hz). The electrode pellets were filled with TD-246 skin conductance electrode paste (Med. Assoc. Inc., St. Albans, VT, USA) and attached to the thenar and hypothenar eminences of a participant's left hand. EMG data were rectified and exported together with EDA data at a sampling interval of 0.49 ms to SPSS (SPSS Inc., Chicago, IL, USA) for further analysis. Data were considered to be invalid when no signal was recorded for long periods (e.g., electrode fell off or equipment error). These data were excluded from further analysis: this was the case for seven participants.

Different components of game experience were measured using the gameplay experience questionnaire GEQ. It combines several game-related subjective measures (with a total of 36 questions): immersion, tension, competence, flow, negative affect, positive affect and challenge. Each dimension has five items (except immersion which has six items). Each item consists of a statement on a five-point scale ranging from 0 (not agreeing with the statement) to 4 (completely agreeing with the statement). Example statements are “I forgot everything around me” (Flow), “I was good at it” (Competence), “I felt that I could explore things” (Immersion), “I felt frustrated” (Tension), “I had to put a lot of effort into it” (Challenge), “I enjoyed it” (Positive Affect), and “I was distracted” (Negative Affect). The questionnaire was developed based on focus group research and subsequent survey studies (Cronbach’s alpha values ranged from .71 to .89 in the original study).

26.5.2 Experimental Design

We employed a 2×2 repeated-measures factorial design using sound (on and off) and music (on and off) as independent variables, using a counter-balanced order of sound and music game-level stimuli. Thus the conditions were: (1) Sound on, Music off, (2) Sound off, Music off, (3) Sound on, Music on, (4) Sound off, Music on. EMG and EDA responses were measured together with questionnaire items indicating the overall game experience for the different playing conditions. Questionnaire item order was randomized for each participant.

Data were recorded from 36 undergraduate students (66.7 %) and University employees. Their age ranged between 18 and 41 ($M=24$, $SD=4.9$). Gender was not evenly distributed, since only 19.4 % of all participants were female. All participants played digital games regularly, and 94.4 % reported they play games at least once a week. 94.4 % believed they had full hearing capacity. 41.7 % saw themselves as hobby musicians, while only 33.3 % played an instrument, which can be explained by people working with sound recording and programming but not playing an instrument. All participants considered sound at least “somewhat important” in games.

Although Half-Life 2 allows the control of game audio features internally, sound and music were controlled externally for this experiment. For example, a music track was triggered externally, which was audible during playing and a software trigger controlled whether the game engine would play game sound or not.

26.5.3 Data Processing

We approached data processing in two different ways. First, we were investigating tonic or cumulative responses of the cumulative time period when playing the game in each of the condition, so after filtering and rectifying the signal, we compared the average values of each individual condition using inferential statistics. Second, we opted for an event-based analysis approach, where data was clustered 7 s around

player death events. We averaged seven 1-s means, 1 s before (baseline; Second 1) and 6 s after the event (the death of the player; Seconds 2–7). To normalize the distributions of physiological data a natural logarithm was taken from EDA and EMG signals. All data were analyzed in SPSS (SPSS Inc., Chicago, IL, USA) by the linear mixed model procedure with restricted maximum likelihood estimation and a first-order autoregressive covariance structure for the residuals. Participant ID was specified as the subject variable, while the game audio conditions (sound on/music off; sound off/music off; sound on/music on; sound off/music on), the sequence number of the event, and second (1–7) were specified as the repeated variables. When examining the main effects of game events, the condition, sequence number of an event, and second were selected as factors, and a fixed-effects model that included the main effects of these variables was specified. When examining the interaction effects of condition and game events on physiological activity, the condition, sequence number of an event, and second were selected as factors, and a fixed-effects model that included the main effects of these variables and the condition \times second interaction was specified.

Main effects of event-related changes in physiological activity were tested using the following contrasts:

- *Contrast 1*: baseline (Second 1) vs. response (Seconds 2–7).
- *Contrast 2*: linear trend across Seconds 1–7.
- *Contrast 3*: quadratic trend across Seconds 1–7.

Interactions were tested for both quadratic and linear trends. However, since the interaction contrasts with quadratic trends yielded no significant associations, only those using linear trends are reported as follows:

- *Interaction Contrast 1a*: sound vs. no sound \times linear trend across Seconds 1–7. *Interaction Contrast 1b*: sound vs. no sound \times change from baseline (Second 1 vs. Seconds 2–7).
- *Interaction Contrast 2a*: music vs. no music \times linear trend across Seconds 1–7. *Interaction Contrast 2b*: sound vs. no sound \times change from baseline (Second 1 vs. Seconds 2–7).
- *Interaction Contrast 3a*: both music and sound vs. neither \times linear trend across Seconds 1–7. *Interaction Contrast 3b*: sound vs. no sound \times change from baseline (Second 1 vs. Seconds 2–7).
- *Interaction Contrast 4a*: only music vs. only sound \times linear trend across Seconds 1–7. *Interaction Contrast 4b*: sound vs. no sound \times change from baseline (Second 1 vs. Seconds 2–7).

26.5.4 Some Findings from the Tonic Analysis

We used a two-way repeated-measures factorial analysis of variance (ANOVA) using sound and music as independent variables with two levels (on=audible or off=inaudible) and facial EMG (brow, eye, cheek), EDA, and GEQ dimensions as

dependent variables. Before the analysis, average values of psychophysiological measures were normalized using logarithmic transformation. Using the tonic data (in this context, we understand tonic as measured over a period of time albeit responding to an experimental condition) we tested significant differences between the factors sound and music. We ran 2×2 ANOVAs for each EMG measure. The ANOVAs showed no statistical differences for EMG measurements. We have to assume that neither sound nor music, nor the interaction of sound and music, had a significant accumulative effect on EMG measurement. No threshold values existed to classify EDA results as either activation or deactivation in the arousal dimension. Higher arousal values could indicate a more exciting experience in any of the conditions. Using a 2×2 ANOVA, we tested the effects of the independent variables, as for the results of EMG, but no significant cumulative effects were found. Thus, we assumed that neither sound nor music, nor the interaction of sound and music, had a significant effect on EDA measurement.

For the questionnaire data, we tested the effects of sound and music with a 2×2 ANOVA and found a main effect of sound on all seven dimensions of the GEQ and an interaction effect of sound and music on tension and flow. Absence or presence of sound influences all subjective GEQ dimensions, but we could not further determine this effect by an interaction of tension and flow. The more positive dimensions of the GEQ (Flow, Positive Affect, Competence, Immersion, Challenge) were rated higher when the sound of the game was playing, while the more negative dimensions (Negative Affect, Tension) were rated lower. When sound was turned off, we could see the opposite effect. Based on these subjective results, game sound is crucial for a subjectively positive gameplay experience.

We also found an interaction of sound \times music on the GEQ dimensions tension and flow. Flow was rated highest when sound was on and music was off, but received the lowest scores when everything was turned off. When sound and music were on, the flow rating was a slightly lower than in the sound on/music off condition. The experience ratings were even lower when sound was turned off and music remained on. Turning on non-diegetic music (that is music that does not directly relate to gameplay) seemed to dampen the flow experience (which was more polarized in positive and negative dimensions) when the differences of sound were taken into account. Regarding tension ratings, when music was on and sound was on, tension was experienced lowest while when music was on and sound was off, tension was rated highest. There was not much difference when music was off.

26.5.5 Some Findings from the Event-Based Analysis

For the event-based analysis, we used a linear mixed model analysis procedure. We found that regardless of the condition, EMG activity for all investigated muscle areas (brow, eye, and cheek) presented a statistically significant quadratic increase. In Condition 1 (sound on, music off), Contrast 1 (first second vs. seconds 2–7) revealed that the response to a death event was a significant increase in EMG activity

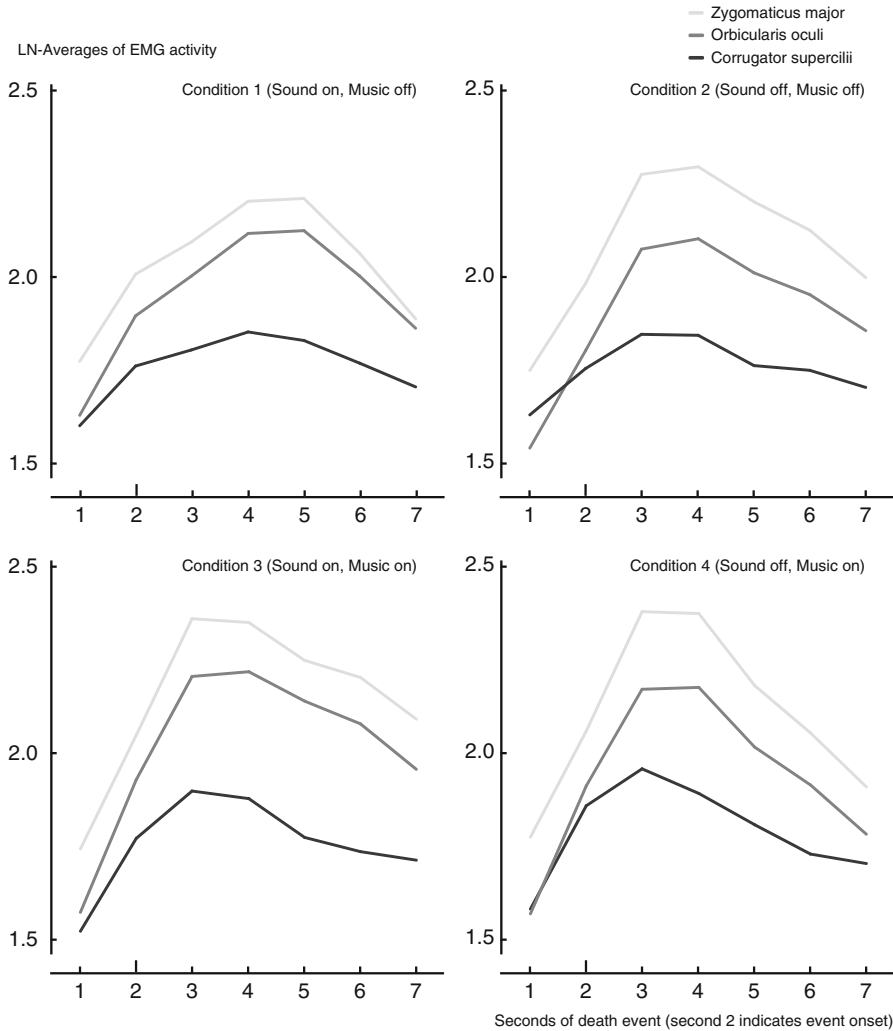


Fig. 26.8 Averages of EMG activity during each of the four conditions for cheek (ZM), eye (OO), and brow (CS) muscles

for CS, OO, and ZM muscle areas ($p < .001$). Contrast 2, testing linear trend, was significant for OO EMG activity ($p = .003$), but not for others.

Results of Contrast 3 showed that the trend was quadratic (first rising and then declining) for all EMG activity measures ($p < .001$, see Fig. 26.8). This tendency was repeated in Condition 2 (both sound and music off), Condition 3 (both sound and music on), and Condition 4 (sound off, music on): Contrast 1 showed that the response was increasing in relation to the baseline second for EMG activity over all muscle areas (all $p < .001$), and Contrast 3 that the response was quadratic, that is, first increasing but decreasing within 7 s (all $p < .001$). Contrast 2 yielded significant

associations in Condition 2, where sound and music were both turned off, for eye EMG ($p < .001$) and cheek EMG activity ($p = .010$), like it did in Condition 3, where both sound and music were turned on, ($p < .001$ and $p = .007$, respectively). In Condition 4 (sound off, music on) none of the Contrast 2 tests showed significant associations. However, in all the cases where Contrast 2 was significant suggesting linear trend, the t -value for contrast 3 was higher, revealing a trend that was predominantly quadratic. Thus, the results of Contrasts 2 and 3 together indicate that in the most cases, the response to the death event is not an increase in long-term EMG activity level, but rather a transitory peak in EMG activity. The response peaked around Second 3 or 4 in all conditions except Condition 1, where the peak occurred approximately 1 s later. In summary, no condition had an effect on the EMG responses elicited by the death event, since the response was significant but similar in all conditions.

The main effects of death events on electrodermal activity showed less uniform responses. In Conditions 1 (sound on, music off) and 3 (sound on, music on), none of the contrasts revealed significant trends; that is, there was no change from the baseline (first second), no linear, and no quadratic trend in response to the event (see Fig. 26.9).

The trend for EDA in Condition 3 appears linear in visual inspection, but because the amount of death events in this condition was lower than in others and it did not quite reach statistical significance ($p = .064$).

In Condition 2 (sound off, music off), both positive linear and negative quadratic trends ($p = .023$ and $.013$ for contrasts 2 and 3, respectively) were found, latter being stronger. Only Contrast 2 showed a significant trend in Condition 4 (sound off, music on) ($p = .031$), revealing a linear increase in EDA as a response to the event.

We tested the interaction between the condition and linear trend of EDA over 7 s using contrast analyses. None of the interactions between condition and quadratic trends showed significant associations. Whereas all interactions using linear and quadratic trend EMG activity were non-significant, Interaction Contrasts 2b and 3b testing change from baseline (Second 1) to response (Seconds 2–7) showed that the brow EMG activity level rose in response to the death event more when music was on than when it was off ($p = .018$), and more when both music and sound were on vs. when they were both off ($p = .017$).

For EDA, Interaction Contrast 1a, testing the interaction of linear trend and the effect of sound vs. no sound, showed that the event prompted a greater linear increase when the sound was off compared to condition where the sound was on (regardless of music). That is, the participants responded with greater arousal to the death event when there were no sounds. Interaction Contrast 2a, testing the interaction of linear trend and the effect of music vs. no music, similarly showed that the event prompted a greater linear increase (greater arousal) when the music was on, as compared to when music was off (regardless of sound). Interaction Contrast 3a was not significant, suggesting that there was no difference in the EDA response whether both music and sound were on or off. Interaction Contrast 4a, testing the interaction of linear trend and the effect of only music vs. only sound, revealed that the event elicited a greater linear increase in EDA when music was on and sound

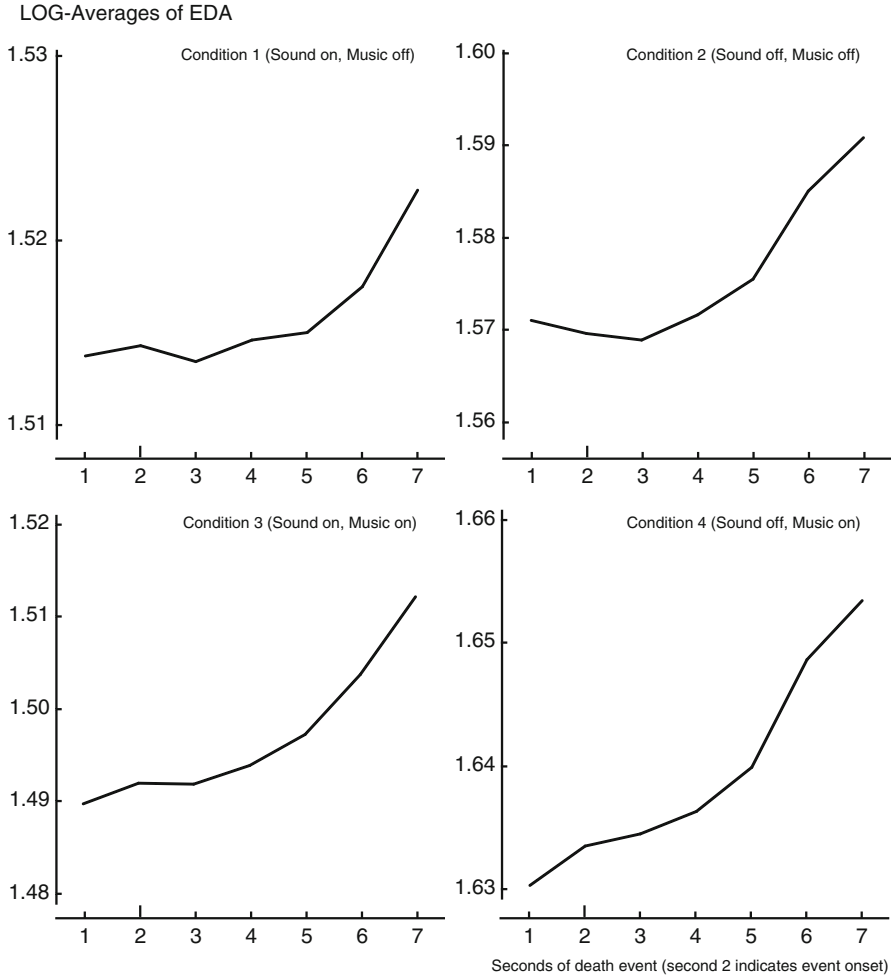


Fig. 26.9 EDA averages during each of the four conditions for cheek (ZM), eye (OO), and brow (CS) muscles. Note the different scales for each condition in the figure

was off, compared to the opposite condition. Interaction Contrast 4b showed that this linear increase was also a significant increase as compared to the baseline (Second 1; $p = .021$). In conclusion, the EDA response to the death event increased when the sound was off or music was on.

26.5.6 Takeaway from the Case Study

This case study demonstrates two different approaches to analyzing physiological data and the different the results that these types of analyses yield. For this particular

study design regarding sound and music in games, we could take away that tonic effects (meaning effects that are measured throughout a game session) can be better gauged using questionnaires (which again emphasizes the importance of using mixed methods in Games User Research). For gameplay related events, in this case death events, physiological measures seem to provide some insights into short physiological experience peaks that seem to occur 3–4s after an event onset. This is particularly interesting since we are currently moving toward mixed-method automatic marker-driven analysis of physiological metrics and gameplay metrics. While the conditions did not yield a significant response, the death event was meaningful to the gameplay in terms of its physiological response. The interpretation of psychophysiological measures and data depends on the context and research paradigm that is being followed. We should evaluate other statistical tests and a non-linear analysis of the relationship between physiology and player experience in the future. For example, applying artificial neural networks to the analysis process might yield some more interesting relationships than the standard inferential tests used here. The interpretation of these measures in a gaming context requires more validation before we can ensure consistent results and guidelines useful to the industry. One approach in this direction has recently been reported in a study by Mirza-Babaei et al. (2013), which addresses the usefulness of biometric measures in a games user research scenario and provides user testing guidelines for games user researchers interested in physiological measures.

26.6 How Can Physiological Metrics Be of Value to the Game Industry?

Given that this is only a brief introduction of psychophysiological measures for the game industry, we also have to discuss in which scenarios physiological game research is useful. Fairclough (2011a, b) suggested a thought experiment, where he outlines ten suggestions for improving the use physiological metrics in game user research. Many of these suggestions should be implemented by game user researchers for physiological metrics to work most effectively in an industry setting.

- **Physiological metrics can be recorded continuously during a game user research session without interrupting play.** This makes these methods superior to subjective measures that either break the experience (by interrupting and prompting with questions) or introduce memory bias (by asking questions about the game in retrospect). The only downside of physiological metrics is that the player has to wear sensors and that some might find this intrusive (although based on personal experience, many players forget that they wear sensors a few minutes into the game).
- **A game user researcher interested in physiological assessment of players needs to be well-informed about what each sensor type measures.** Company executives and the marketing department need to understand this is no emotion quantifier or thought printer. Sensors measure electrical activity that comes from

motor, skin, or brain activity and depending on the area of application allows some conclusion of the activity of the body area being measured. This also means that we need an experience vocabulary working from a high level psychological concept (engagement) toward the low-level body response (sympathetic activation → higher heart rate). Inferences made from low-level body responses to high level concepts in comparison are always difficult to withstand closer scientific inspection.

- **For capturing player experience, a hypothesis-driven approach is suggested**, where only one particular aspect of experience is under investigation. Ideally this aspect is well-defined in related literature so that, for example, we only investigate positive and negative emotional responses to a certain game event or game area or that we investigate cognitive workload during a game tutorial.
- **To establish a link between ideal player experience and the corresponding physiological responses, we should investigate responses to key aspects when naïve participants play the most successful games of the industry** (in terms of financial and critical success). If we could find out what physiological responses relate to the player experiences that drive the success of these games, we could work towards establishing a physiological success metric. This would be truly valuable for the game industry.
- **In every aspect of physiological experimentation we need to be aware that the human body is still present in the real world while playing a video game.** Our nervous system therefore responds to real-world stimulation coming from our environment as well as to cognitive stimulation. These contextual influences (that may be overlooked during the screening of a participant) can result in changes in emotion or motivation during the experiment. Influences such as room temperature, movement, drugs, chemicals, noise, and many more can also introduce contextual bias into our interpretation of physiological activity. In the end, it is important to keep in mind how sensitive our nervous system really is when interpreting physiological metrics.
- **Physiological metrics do not distinguish between physical activity and psychological events.** Three components are involved in recording physiological metrics: external physical activity, internal emotional activity, and internal cognitive activity (Stemmler et al. 2007).
- **Given what we now know about physiological responses, we will always have a certain signal-to-noise ratio in our physiological metrics.** We can counteract the amount of noise by enforcing a strict experimental protocol in a very controlled environment or by recording all possible confounds with additional sensors (e.g., temperature, noise, light) to remove their influence during analysis.
- **Before testing players, it is important to carefully record their demographic background**, including their skill level and past game preferences and experiences. Novelty and habituation can impact physiological responses considerably.

- **It is important to create the different experimental conditions carefully** within a systematically manipulated environment (e.g., a game engine). Ideally, we only change one variable at a time.
- **Metrics should be tracked together.** Other gameplay tracking metrics can be considered overt behavior markers in the game world as they are visible instantly whereas physiological metrics are covert measures that are not always visible directly. Both metrics should be tracked together and a possible relationship between them should be explored using statistical analyses. Subjective responses are best recorded after physiological measurement.

We can conclude that psychophysiological measures in games should not be used alone, but always in conjunction with other measures to establish relationships between player experience and physiological responses. Much work remains to be done in this area before it becomes part of the everyday testing of game user researchers. However, given recent advances by sensor manufacturers, this technology will eventually be more common in game user research. When we start using it to improve our games, we will always need to remember its sensitivity and the possible contextual influences, so that all interpretations should be understood with a grain of salt.

26.7 Next Steps in Using Physiological Metrics

Imagine you would like to use these measures in your company, a good way to get started with most of them would be to consider that if you are measuring people, you always have to account for individual differences and you have to be very clear on your research goal. Can you obtain a result for your research question in an easier and more cost-effective way? If your answer is yes, then physiological measures might not be for you at this point. I would suggest the easiest way to get started with psychophysiological measures would be to obtain or build a system to measure electrodermal response, not only because it has a direct relationship to arousal or excitement, but also because this type of data can be easily processed and analyzed. We are currently working on methods to make the interpretation of electrodermal activity within a gaming context easier to interpret and more accessible for the games industry (e.g., using biometric storyboards) (Mirza-Babaei 2011; Mirza-Babaei et al. 2012).

Another option that we have worked with in the past and that we will continue working on in the future is building physiological measurement systems that track physiological metrics and player events together (Nacke et al. 2008; Kivikangas et al. 2011b). For these systems, a few key issues have to be considered:

- Synchronized time data. Often the timestamp is used as the identification key to merge different log data. If you are logging physiological data on a different system than your other metrics, make sure your timestamp is synchronized across the network.

- Data storage. Physiological log files become large over time (similar to video recording files). It is a good idea to have a backup or extended storage solution in mind when starting to seriously collect physiological metrics data.
- Choose the right gameplay hooks. Depending on your analysis, you will need to make a decision what gameplay hooks are relevant in your log data to be related to physiological measures.
- Set up a demo version of your game where players can focus on one action at a time. This will make your psychophysiological analysis much easier since you are concentrating on one key variable.

If you plan to use more elaborate physiological metrics, such as EEG, an expert opinion is often helpful to get started. EEG is hard to interpret and there is still not enough evidence to show its usefulness for researching interesting game situations. In the end all the research needs to tie the ideas back to improving game design. EEG has a lot of potential to investigate meaningful decisions at certain key points in your game and once we have better signal processing and automated analysis techniques, it will likely see a larger adoption in the game industry.

This chapter might have shown to you that physiological evaluation is a field that requires some consideration and is not as simple just sticking a sensor on persons and getting precise emotional readouts of their activities. Therefore, I can only recommend considering the goal of your game user research study before you decide on whether or not you can use sensors to solve your problem. Physiological sensors provide a great addition to other quantitative game metrics, because of their potential for adding emotional meaning to your data. If you want a complete and robust picture of the user experience when playing games, you should consider adding physiological measures to your quantitative toolbox since they will provide rich evaluation possibilities over time.

Acknowledgements Thanks to Pejman Mirza-Babaei for his initial feedback and Anders Drachen for his encouragement and feedback on this chapter. This research was supported by the Network of Centres of Excellence (NCE), Graphics, Animation and New Media (GRAND), NSERC, and SSHRC IMMERSe (895-2011-1014).

Additional Reading for Psychophysiology

1. Andreassi, J. L. (2000). Human behavior and physiological response. *Psychophysiology* 4, 1–12.
2. Hazlett, R. L., & Benedek, J. (2007). Measuring emotional valence to understand the user's experience of software. *International Journal of Human-Computer Studies* 65(4), 306–314. doi:[10.1016/j.ijhcs.2006.11.005](https://doi.org/10.1016/j.ijhcs.2006.11.005)
3. Baumgartner, T., Valko, L., Esslen, M., & Jäncke, L. (2006). Neural correlate of spatial presence in an arousing and noninteractive virtual reality: An EEG and psychophysiology study. *CyberPsychology & Behavior* 9(1), 30–45. doi:[10.1089/cpb.2006.9.30](https://doi.org/10.1089/cpb.2006.9.30)
4. Hudlicka, E. (2003). To feel or not to feel: The role of affect in human–computer interaction. *International Journal of Human-Computer Studies* 59(1–2), 1–32. doi:[10.1016/s1071-5819\(03\)00047-8](https://doi.org/10.1016/s1071-5819(03)00047-8)

5. Hoffman, J. E. (1998). Visual attention and eye movements. In H.E. Pashler (Ed.) *Attention*. New York: Psychology Press (Taylor & Francis).
6. Fried, R. (1982). On-line analysis of the GSR. *Integrative Physiological and Behavioral Science* 17(2), 89–94. doi:[10.1007/bf03002004](https://doi.org/10.1007/bf03002004)

About the Author

Dr. Lennart E. Nacke is an Assistant Professor for Human-Computer Interaction and Game Science in the Faculty of Business and Information Technology at the University of Ontario Institute of Technology (UOIT) in Canada and currently also a visiting lecturer at the Department of Informatics at the University of Sussex. He is working on projects that deal with the cognitive and emotional side of playing games. His goal is to describe, analyse and have computers and machines react to the thoughts and feelings you have when you are fully engaged in playing a video game. With this research we can start understanding the effects of video games (positive and negative). Using body sensor technology, for example, sensors for brainwaves, heart rate, or muscle tension, Dr. Nacke taps into some of video gaming's most motivating features to improve our physical fitness and mental wellbeing. His research is funded nationally and internationally and his publications have won best paper honourable mention (awarded to the top 5%) and best paper awards (awarded to the top 1%) at the premier human-computer interaction conferences CHI 2011 and CSCW 2012. He also writes a blog at www.acagamic.com and you can find out more about his research group at <http://hcigames.businessandit.uoit.ca>

References

- Andreassi, J. L. (2006). *Psychophysiology: Human behavior and physiological response*. London: Routledge.
- Barrett, L. F. (2006). Are emotions natural kinds? *Perspectives on Psychological Science*, 1(1), 28–58. doi:[10.1111/j.1745-6916.2006.00003.x](https://doi.org/10.1111/j.1745-6916.2006.00003.x).
- Barrett, L. F., Mesquita, B., Ochsner, K. N., & Gross, J. J. (2007). The experience of emotion. *Annual Review of Psychology*, 58(1), 373–403. doi:[10.1146/annurev.psych.58.110405.085709](https://doi.org/10.1146/annurev.psych.58.110405.085709).
- Boucsein, W. (1992). *Electrodermal activity*. New York: Plenum Press.
- Bradley, M. M., & Lang, P. J. (2007). Emotion and motivation. In J. T. Cacioppo, L. G. Tassinary, & G. G. Berntson (Eds.), *Handbook of psychophysiology* (3rd ed., pp. 581–607). New York: Cambridge University Press.
- Bradley, M. M., Codispoti, M., Cuthbert, B. N., & Lang, P. J. (2001). Emotion and motivation I: Defensive and appetitive reactions in picture processing. *Emotion*, 1(3), 276–298.
- Brathwaite, B., & Schreiber, I. (2008). *Challenges for game designers*. Boston: Charles River Media.
- Brave, S., & Nass, C. (2002). Emotion in human-computer interaction. In J. A. Jacko & A. Sears (Eds.), *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications* (pp. 81–96). Mahwah: Lawrence Erlbaum Associates.
- Cacioppo, J. T., Gardner, W. L., & Berntson, G. G. (1999). The affect system has parallel and integrative processing components: Form follows function. *Journal of Personality and Social Psychology*, 76(5), 839–855. doi:[10.1037/0022-3514.76.5.839](https://doi.org/10.1037/0022-3514.76.5.839).

- Cacioppo, J. T., Tassinary, L. G., & Berntson, G. G. (Eds.). (2007). *Handbook of Psychophysiology*. Cambridge, UK: Cambridge University Press.
- Calvo, R. A., & D'Mello, S. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. *IEEE Transactions on Affective Computing*, 1(1), 18–37. doi:10.1109/t-affc.2010.1.
- Cannon, W. B. (1927). The James-Lange theory of emotions: A critical examination and an alternative theory. *The American Journal of Psychology*, 39(1/4), 106–124. doi:10.2307/1415404.
- Chatrino, G. E., Lettich, E., & Nelson, P. L. (1988). Modified nomenclature for the “10%” electrode system. *Journal of Clinical Neurophysiology: Official Publication of the American Electroencephalographic Society*, 5(2), 183–186.
- Dalgleish, T., Power, M. J., & Power, M. (2000). *Handbook of cognition and emotion*. Chichester/ New York: Wiley.
- Dalgleish, T., Dunn, B. D., & Mobbs, D. (2009). Affective neuroscience: Past, present, and future. *Emotion Review*, 1(4), 355–368. doi:10.1177/1754073909338307.
- Damasio, A. R. (1994). *Descartes' error*. New York: G.P. Putnam.
- Darrow, C. W. (1964). Psychophysiology, yesterday, today, and tomorrow. *Psychophysiology*, 1(1), 4–7. doi:10.1111/j.1469-8986.1964.tb02615.x.
- Davidson, R. J., Scherer, K. R., & Goldsmith, H. H. (2003). *Handbook of affective sciences*. Oxford: Oxford University Press.
- Döveling, K., Von Scheve, C., & Konijn, E. A. (2010). *The Routledge handbook of emotions and mass media*. London: Routledge.
- Duchowski, A. T. (2007). *Eye tracking methodology: Theory and practice* (2nd ed.). New York: Springer.
- Ekman, P. (1972). Universals and cultural differences in facial expressions of emotion. In J. Cole (Ed.), *Nebraska symposium on motivation, 1971* (pp. 207–283). Lincoln: University of Nebraska Press.
- Ekman, P. (1992a). Are there basic emotions? *Psychological Review*, 99(3), 550–553.
- Ekman, P. (1992b). An argument for basic emotions. *Cognition and Emotion*, 6(3/4), 169–200. doi:10.1080/02699939208411068.
- Ekman, P., & Davidson, R. J. (1994). *The nature of emotion: Fundamental questions*. New York: Oxford University Press.
- Engl, S., & Nacke, L. E. (2013, February). Contextual influences on mobile player experience – A game user experience model. *Entertainment Computing* 4(1), 83–91, ISSN 1875-9521. doi:10.1016/j.entcom.2012.06.001.
- Fairclough, S. H. (2011a). Biometrics and evaluation of gaming experience part two: A thought experiment. <http://www.physiologicalcomputing.net/?p=1760>. Last accessed 19 Mar 2013.
- Fairclough, S. H. (2011b). Biometrics, game evaluation, and user XP: Approach with caution. <http://www.physiologicalcomputing.net/?p=1698>. Last accessed 19 Mar 2013.
- Fridlund, A. J., & Cacioppo, J. T. (1986). Guidelines for human electromyographic research. *Psychophysiology*, 23(5), 567–589. doi:10.1111/j.1469-8986.1986.tb00676.x.
- Frijda, N. H. (1986). *The emotions*. Cambridge, UK: Cambridge University Press.
- Gokcay, D., & Yildirim, G. (2010). *Affective computing and interaction: Psychological, cognitive, and neuroscientific perspectives*. Hershey: IGI Global.
- Gualeni, S., Janssen, D., & Calvi, L. (2012). How psychophysiology can aid the design process of casual games: A tale of stress, facial muscles, and paper beasts. In *Proceedings of the international conference on the Foundations of Digital Games (FDG '12)* (pp. 149–155). Raleigh: ACM. doi:10.1145/2282338.2282369.
- Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock & N. Meshkati (Eds.), *Human mental workload* (pp. 239–250). Amsterdam: North Holland Press.
- Hazlett, R. L. (2006). Measuring emotional valence during interactive experiences: Boys at video game play. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '06)* (pp. 1023–1026). Montréal: ACM. doi:10.1145/1124772.1124925.

- Hazlett, R. L. (2008). Using biometric measurement to help develop emotionally compelling games. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advancing the player experience* (pp. 187–205). Burlington: Morgan Kaufmann.
- Hugdahl, K. (1995). *Psychophysiology: The mind-body perspective*. Cambridge: Harvard University Press.
- Izard, C. E. (1972). *Patterns of emotions: A new analysis of anxiety and depression*. New York: Academic.
- James, W. (1884). What is an emotion? *Mind*, 9(34), 188–205.
- Jasper, H. H. (1958). Report of the committee on methods of clinical examination in electroencephalography. *Electroencephalography and Clinical Neurophysiology*, 10, 370–375.
- Kivikangas, J. M., Chanel, G., Cowley, B., Ekman, I., Salminen, M., Järvelä, S., & Ravaja, N. (2011a). A review of the use of psychophysiological methods in game research. *Journal of Gaming & Virtual Worlds*, 3(3), 181–199. doi:10.1386/jgvw.3.3.181_1
- Kivikangas, J. M., Nacke, L., & Ravaja, N. (2011b). Developing a triangulation system for digital game events, observational video, and psychophysiological data to study emotional responses to a virtual character. *Entertainment Computing*, 2(1), 11–16. doi:10.1016/j.entcom.2011.03.006.
- Kleinginna, P. R., & Kleinginna, A. M. (1981). A categorized list of emotion definitions, with suggestions for a consensual definition. *Motivation and Emotion*, 5(4), 345–379.
- Lacey, J. I. (1959). Psychophysiological approaches to the evaluation of psychotherapeutic process and outcome. In E. A. Rubinstein & M. B. Parloff (Eds.), *Research in psychotherapy* (pp. 160–208). Washington, DC: American Psychological Association. doi:10.1037/10036-010.
- Lane, R. D., & Nadel, L. (2002). *Cognitive neuroscience of emotion*. New York: Oxford University Press.
- Lang, P. J. (1995). The emotion probe. Studies of motivation and attention. *American Psychologist*, 50, 372–385.
- Lange, C. G. (1912). The mechanism of the emotions. In B. Rand (Ed.), *The classical psychologists* (pp. 672–684). Boston: Houghton Mifflin.
- Larsen, J. T., McGraw, A. P., & Cacioppo, J. T. (2001). Can people feel happy and sad at the same time? *Journal of Personality and Social Psychology*, 81(4), 684–696. doi:10.1037/0022-3514.81.4.684.
- Lazarus, R. S. (1968). Emotions and adaptation: Conceptual and empirical relations. In *Nebraska symposium on motivation*. Lincoln: University of Nebraska Press.
- LeDoux, J. (1998). *The emotional brain*. London: Orion Publishing Group.
- Lewis, M., Haviland-Jones, J. M., & Barrett, L. F. (2010). *Handbook of emotions* (3rd ed.). New York: Guilford Press.
- Lykken, D. T., & Venables, P. H. (1971). Direct measurement of skin conductance: A proposal for standardization. *Psychophysiology*, 8(5), 656–672. doi:10.1111/j.1469-8986.1971.tb00501.x.
- Mandryk, R. (2008). Physiological measures for game evaluation. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advice from the experts for advancing the player experience* (pp. 207–235). Burlington: Morgan Kaufmann.
- Mandryk, R. L., & Atkins, M. S. (2007). A fuzzy physiological approach for continuously modeling emotion during interaction with play environments. *International Journal of Human Computer Studies*, 65(4), 329–347. doi:10.1016/j.ijhcs.2006.11.011.
- Mandryk, R. L., Inkpen, K. M., & Calvert, T. W. (2006). Using psychophysiological techniques to measure user experience with entertainment technologies. *Behavior & Information Technology*, 25(2), 141–158. doi:10.1080/01449290500331156.
- Mirza-Babaei, P. (2011). Understanding the contribution of biometrics to games user research. In *Proceedings of “Think Design Play: The fifth international conference of the Digital Research Association” (DIGRA 2011)*, Hilversum. http://www.digra.org/dl/display_html?chid=11310.43254.pdf. Last accessed 19 Mar 2013.
- Mirza-Babaei, P., Nacke, L. E., Fitzpatrick, G., White, G., McAllister, G., & Collins, N. (2012). Biometric storyboards: Visualising game user research data. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts (CHI EA '12)* (pp. 2315–2320). Austin: ACM. doi:10.1145/2223656.2223795.

- Mirza-Babaei, P., Nacke, L., Gregory, J., Collins, N., & Fitzpatrick, G. (2013). How does it play better? Exploring user testing and biometric storyboards in games user research. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI 2013)*. To appear.
- Nacke, L. E. (2009). *Affective ludology: Scientific measurement of user experience in interactive entertainment*. Ph.D. thesis, Blekinge Institute of Technology, Karlskrona, Sweden.
- Nacke, L., & Lindley, C. A. (2008). Flow and immersion in first-person shooters: Measuring the player's gameplay experience. In *Proceedings of the 2008 conference on future play: Research, play, share (Future Play '08)* (pp. 81–88). Toronto: ACM. doi:[10.1145/1496984.1496998](https://doi.org/10.1145/1496984.1496998).
- Nacke, L., Lindley, C., & Stellmach, S. (2008). Log who's playing: Psychophysiological game analysis made easy through event logging. In P. Markopoulos, Bd. Ruyter, W. IJsselsteijn, D. Rowland (Eds.), *Proceedings of Fun and Games, Second International Conference*, Eindhoven, The Netherlands, October 20–21, 2008 (Lecture Notes in Computer Science) (pp. 150–157). Dordrecht: Springer. doi:[10.1007/978-3-540-88322-7_15](https://doi.org/10.1007/978-3-540-88322-7_15)
- Nacke, L. E., Grimshaw, M. N., & Lindley, C. A. (2010). More than a feeling: Measurement of sonic user experience and psychophysiology in a first-person shooter game. *Interacting with Computers*, 22(5), 336–343. doi:[10.1016/j.intcom.2010.04.005](https://doi.org/10.1016/j.intcom.2010.04.005).
- Nacke, L. E., Kalyn, M., Lough, C., & Mandryk, R. L. (2011). Biofeedback game design: Using direct and indirect physiological control to enhance game interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI '11)* (pp. 103–112). Vancouver: ACM. doi:[10.1145/1978942.1978958](https://doi.org/10.1145/1978942.1978958).
- Nishimoto, S., Vu An, T., Naselaris, T., Benjamini, Y., Yu, B., & Gallant Jack, L. (2011). Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology: CB*, 21(19), 1641–1646.
- Panksepp, J. (2004). *Affective neuroscience: The foundations of human and animal emotions*. Oxford, UK: Oxford University Press.
- Pelachaud, C. (2012). *Emotion-oriented systems*. Hoboken: Wiley.
- Picard, R. W. (1997). *Affective computing*. Cambridge, MA: MIT Press.
- Plutchik, R. (1980). A general psychoevolutionary theory of emotion. In R. Plutchik & H. Kellerman (Eds.), *Emotion: Theory, research, and experience* (Vol. 1, 3, pp. 3–31). New York: Academic.
- Plutchik, R. (1991). *The emotions*. Lanham: University Press of America.
- Prokasy, W. F., & Raskin, D. C. (1973). *Electrodermal activity in psychological research*. New York: Academic.
- Ravaja, N. (2004). Contributions of psychophysiology to media research: Review and recommendations. *Media Psychology*, 6(2), 193–235. doi:[10.1207/s1532785xmep0602_4](https://doi.org/10.1207/s1532785xmep0602_4).
- Ravaja, N., Turpeinen, M., Saari, T., Puttonen, S., & Keltikangas-Järvinen, L. (2008). The psychophysiology of James Bond: Phasic emotional responses to violent video game events. *Emotion*, 8(1), 114–120. doi:[10.1037/1528-3542.8.1.114](https://doi.org/10.1037/1528-3542.8.1.114).
- Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6), 1161–1178. doi:[10.1037/h0077714](https://doi.org/10.1037/h0077714).
- Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological Review*, 110(1), 145–172. doi:[10.1037/0033-295x.110.1.145](https://doi.org/10.1037/0033-295x.110.1.145).
- Sander, D., & Scherer, K. R. (2009). *The Oxford companion to emotion and the affective sciences*. Oxford/New York: Oxford University Press.
- Schachter, S. (1964). The interaction of cognitive and physiological determinants of emotional state. In L. Berkowitz (Ed.), *Advances in experimental social psychology* (Vol. 1, pp. 49–80). New York: Academic.
- Schachter, S., & Singer, J. (1962). Cognitive, social, and physiological determinants of emotional state. *Psychological Review*, 69, 379–399.
- Scherer, K. R. (1984). Emotion as a multicomponent process: A model and some cross-cultural data. *Review of Personality & Social Psychology*, 5, 37–63.
- Scherer, K. R., Bänziger, T., & Roesch, E. B. (2010). *Blueprint for affective computing: A source-book*. Oxford/New York: Oxford University Press.
- Smith, M. E., Gevins, A., Brown, H., Karnik, A., & Du, R. (2001). Monitoring task loading with multivariate EEG measures during complex forms of human-computer interaction. *Human*

- Factors: The Journal of the Human Factors and Ergonomics Society*, 43(3), 366–380. doi:[10.1518/001872001775898287](https://doi.org/10.1518/001872001775898287).
- Stemmler, G., Aue, T., & Wacker, J. (2007). Anger and fear: Separable effects of emotion and motivational direction on somatovisceral responses. *International Journal of Psychophysiology*, 66(2), 141–153. doi:[10.1016/j.ijpsycho.2007.03.019](https://doi.org/10.1016/j.ijpsycho.2007.03.019).
- Stern, R. M., Ray, W. J., & Quigley, K. S. (2001). *Psychophysiological recording* (2nd ed.). New York: Oxford University Press.
- Tassinary, L. G., Cacioppo, J. T., & Vanman, E. J. (2000). The skeletomotor system: Surface electromyography. *Handbook of Psychophysiology*, 2, 274–278.
- Waterink, W., & van Boxtel, A. (1994). Facial and jaw-elevator EMG activity in relation to changes in performance level during a sustained information processing task. *Biological Psychology*, 37(3), 183–198. doi:[10.1016/0301-0511\(94\)90001-9](https://doi.org/10.1016/0301-0511(94)90001-9).
- Watson, D., Wiese, D., Vaidya, J., & Tellegen, A. (1999). The two general activation systems of affect: Structural findings, evolutionary considerations, and psychobiological evidence. *Journal of Personality and Social Psychology*, 76(5), 820–838.
- Yannakakis, G. N., & Hallam, J. (2008). Entertainment modeling through physiology in physical play. *International Journal of Human Computer Studies*, 66(10), 741–755. doi:[10.1016/j.ijhcs.2008.06.004](https://doi.org/10.1016/j.ijhcs.2008.06.004).
- Yannakakis, G. N., Hallam, J., & Lund, H. H. (2008). Entertainment capture through heart rate activity in physical interactive playgrounds. *User Modeling and User-Adapted Interaction*, 18(1), 207–243.

Chapter 27

Improving Gameplay with Game Metrics and Player Metrics

Graham McAllister, Pejman Mirza-Babaei, and Jason Avent

Take Away Points:

1. Practical case study on how industry and research can integrate.
2. Example of a combined metrics approach (game metrics and player metrics).
3. Introduction of a new method, which researchers can build upon, i.e. dissecting gameplay via sequential analysis and correlating the behaviors with physiological responses.
4. A new gameplay visualization method called Biometric Storyboards.
5. Using both sets of metrics for player profiling.

27.1 Introduction

Designing and developing video games is typically a long and demanding process. The overall aim of developing a game that is enjoyable and rewarding to play for everyone is a complex one due to the diversity of players who may potentially interact with the game. Understanding how players interact and behave during gameplay is of vital importance to developers. An accurate understanding of the gameplay experience during development can help identify and resolve any potential problem areas

G. McAllister (✉)

Player Research, 38 Holland Road, Hove, East Sussex, BN31JL, UK
e-mail: graham@playerresearch.com

P. Mirza-Babaei

University of Sussex, Brighton BN1 9QU, UK
e-mail: pejman.m@acm.org

J. Avent

Disney Interactive Studios, Glendale, CA 91201, USA

before release, leading to a better player experience and arguably, greater game review scores and sales. There are two main sources from which potentially useful data can be extracted: the video game (game metrics), and the player (player metrics).

As emphasized within several chapters in this book, game metrics (telemetry) have become an increasingly important topic in recent years. They have discussed how telemetry data is captured and analyzed to investigate various aspects of player performance, including progress through the game world (e.g. time taken, location of death) or to balance gameplay. However, as emphasized in Chaps. 17, 18, and 19, in-game telemetry focuses on what players do (their behavior), but does not completely address the player experience issues of ‘why they did it’, or ‘how they felt’. To gather such information, we need to obtain data from the player. While Chaps. 17, 18, and 19, discussed the use of qualitative observations, interviews and questionnaires to gather this data, in this chapter we explore the use of biometrics, physiological sensors attached to the player as a means of capturing unconscious and continuous data. Although physiological sensors have been used by academics in the past to explore players’ reaction to gameplay, they have largely been under-used to date by industry. One notable exception is Valve,¹ who has openly discussed how they use physiological sensors to enhance gameplay (Ambinder 2011).

This chapter presents a case study on Pure (Disney Interactive Studios 2008), an off-road racing game developed by Black Rock Studio² (BRS), which is part of Disney Interactive. The game performed well both critically and commercially, with a rating of 85% on Metacritic.³ The chapter will present an interview with Pure’s game director, focusing on how they used game metrics to adjust difficulty balancing, and also how video game user research studio, Player Research,⁴ used player metrics to identify positive and negative gameplay issues.

In addition to this interview, this chapter will also discuss three new methods relating to player metrics. We will describe their usage by way of a practical case study. The three methods cover the areas of:

1. Identifying key gameplay moments
2. Deconstructing gameplay
3. Visualizing the player experience

Combined, these player metrics approaches offer insights into the player *experience*, and are ideal complements to game metrics approaches which focus on player *behavior*. Before detailing these player metrics methods, however, we shall first discuss how game metrics were used during Pure’s development via an interview with the game director.

¹ <http://www.valvesoftware.com/>

² <http://www.disney.co.uk/disneyinteractivestudios/blackrockstudio/>

³ <http://www.metacritic.com/game/xbox-360/pure>

⁴ <http://www.playerresearch.com>

27.2 Case Study – Part 1 – Interview with Jason Avent

Developed by Black Rock Studios and released for Xbox, PS3 and PC in September 2008, *Pure* is an off-road quad-bike game. The player races against AI (or other players online), and performs tricks which earns them ‘juice’, which they can then exchange for boost to help them gain a higher position in the race. The game is perhaps best known for its huge jumps off of cliffs, giving the player spectacular vistas of the environment whilst hopefully giving a sense of excitement.

We interviewed the game director, Jason Avent, on the use of game metrics in *Pure*.

Q. Was this the first time you’d decided to capture metrics from a game? If so, what made you do it this time?

It occurred to me a while ago, but *Pure* was the first project on which we had time to plan for data capture. This was really the first project we’d worked on that user testing was deployed en masse, too. The two things kind of tied together well.

To answer your question, let me share this story. I have a younger brother. I’m 2 years older than him. This meant that when we grew up together I had the luxury of being better at pretty much everything than he was. For much of the time, this was great, but not always. On rainy days, after we got bored with our toys and finished all of our videogames (or got as far as we could), we’d often turn to multi-player games as a way of passing the time. It’s pretty monotonous if you win every time though, just because you’ve got 2 years more brain development, experience and manual dexterity than your competition. It’s even worse when you’re on the receiving end and get continuously beaten. So, like a big brother, I’d let him win, sometimes. It was the only way I could motivate him to keep playing. Over the days, weeks and months, I became pretty good at balancing the difficulty. If I made it too hard, he’d get frustrated and throw a strop. If I made it too easy, he’d see through the sham, and realize I was letting him win. This led to an equal number of arguments! So there was a very fine balance. But with some games, a curious thing happened. I’m still not sure whether it was purely psychological or not, but I cannot deny it. He got better than me. This really hurt. It got to the point, where I couldn’t beat him, and he would laugh and gloat. Now I look back at it, he was probably just reflecting back my very own behavior at me. I think that what happened was that I learned how to lose, while he was learning how to win. By the time I changed my behavior back to improve my tactics, he had enough time to develop a lot more skill. He also had the psychological boost of recent wins to power him on. Usually, eventually I would find my groove again and start winning consistently. At this point, I’d need to repeat the process so he didn’t get frustrated.

It occurred to me that this is the perfect learning curve. Now I’m a game designer, it also occurs to me that computers are powerful enough now to automate this task perfectly. So that’s what we set out to do. Throw away easy, medium and hard difficulty settings, and instead have a dynamic difficulty setting powered by player performance.

Q. How did you decide which metrics to capture? Did this change over time?

Deciding what to watch is the key. Since the goal was fairly clear and focused, it was relatively easy to find a place to start. It also helped that we were working on a racing game. Essentially, a racing game is very repetitive. You do laps of a track until you win. Also, the game structure of Pure was quite simple and linear too. So we decided that the best performance indicator was whether a player won a race or not. From there we looked at the amount of times a player retried an event before they won. Then we realized that there were hard retries – actually finishing the race; and soft retries – players decided their performance was not good enough before the end of the race and restarted the game. When a player was really focused on nailing first place, they tend to soft restart more. Players could progress through the game structure merely by getting 3rd place or less in some cases. All, but hardcore completionists, tended to do this in the early part of their time with the game.

As we made the game respond to the data dynamically, we had to capture more data to keep the system working. It was necessary to record race times and then individual lap times to get a finer handle on the player's performance. As we made the AI better, players would lose more. Although that then made the AI slower eventually, the feedback loop was not short enough.

Q. How effective were the metrics? What were they good at answering? What were they not so good at?

The metrics were very good at telling us the range of performance that the players were spread over. This really helped us decide on the best and worst AI settings. They also showed us the time it took for players to improve and by how much they improved by the time they finished the game. The metrics never really told us what design decisions to make. They merely provided us with good reasons for making certain judgments that were the basis of our decisions. The metrics also helped us identify more metrics to record! That sounds a bit daffy but by the end of the game's development we had measures, which included tricks, boost, score and jump stats – all of which helped us balance the game, and create a complex dynamic difficulty balancing system.

Q. What unforeseen problems did you run into when capturing/using the metrics?

We learned that races are a lot more than just lap times, and that if you 'fix' one system, you have to take the responsibility of stage managing the whole show. We found that by changing the AI such that you won when you ought to have won and lost when you ought to lose, players were able to see through this, and races became predictable. Thus, we had to direct each race to hide this behavior, making behaviors less predictable and more dynamic. We invented 'race stories'. The first of which split the pack of 15 other racers into three groups: 5 slow guys, 5 medium guys and 5 faster guys. On lap one, you are usually within the back marker group. If you fought your way to the middle, by the second lap, then you had a good chance of winning. If you make your way to the fast pack within the second lap, you have a better chance of winning. The AI followed specific rules like this to make the race interesting by responding to the player's performance dynamically. We wanted to always have people right on your tail, to give you someone to aim for up ahead. It worked pretty well.

Q. How would you change the metrics capture approach for future games?

Reading some of the reviews and forum feedback, it was clear that some players found it too easy, while others found it too hard, so there were clearly still issues with the final game's difficulty. We tested about a 100 people in all to prove that our system worked, but although that's a large commitment of time and effort, 100 people is a tiny fraction of our final user base. There are millions of copies of Pure out there across PS3 and Xbox. If we had the time it would have been good to track data after launch, and provide regular patch updates in response to that data. Much like how social networks game companies do now, we should have leveraged our user base to further balance and improve our game.

Metrics are crucial in modern game development to help with the usability and general design process. They can be used to solve a great many design problems. They take away a lot of unnecessary design discussions, so you can focus on the features and decisions that are more subjective.

I think that for this particular aspect of the game, we fell in love with the idea that metrics and a clever system can be the whole answer to an age-old problem – that of balancing a game's difficulty for many users of various ability. With hindsight, a better solution would have been a blend of the old and the new. The dynamic difficulty system worked well almost all of the time, however, it took away player choice. Some players don't want to be challenged, they just want to enjoy the ride. Other players want a series of static challenges that they can work over and over to beat. An alternative solution, I would have liked to try is to offer up these three difficulty levels: Easy, Dynamic and Hard. Perhaps only offering hard mode when the player completed the game in the Dynamic mode.

27.3 Player Metrics

As it can be seen from the previous sections, game metrics can be very useful for logging players' behaviors. However, they are not so useful at explaining this behavior or the resulting player experience. To help explore the rationale and motivations behind player behavior, we need to use appropriate methods to acquire this data from the player. On its own, game metrics can identify some potential issues with a game, but for the game developers to know what to change in order to improve the game, we should aim to understand *why* it was an issue. This is, for us, the primary aim of player metrics. Player metrics, as used in this chapter, can be considered as any data, which is derived or collected from the player as a physical entity directly, as opposed to the player's gameplay (game metrics), which are collected from the game.

Video games are designed to deliver engaging player experiences covering a wide part of the available emotional spectrum. Sculpting this blend of emotions is the job of the game designer. However, evaluating how close the final game is to the original designer's intent, can be a difficult task. If we were evaluating a website, then this would be where usability testing comes in. A set of tasks would be drawn

up, and the users' ability to complete these tasks would be evaluated using appropriate methods. For games, however, our aim is to understand the player experience, and there are (at least) four key aspects to understanding players and gameplay:

- Behavior – What did the player do?
- Rationale – Why did they behave as they did?
- Perception – What do they think happened?
- Experience – How did this make them feel?

By understanding the relationship between players' behaviors and experience, we can begin to gain better insights into the complex area of understanding the player experience. However, collecting player metrics using traditional methods can lead to problems.

27.3.1 What Is Wrong Today?

Firstly, the usual methods of determining the players' experience, such as questionnaires and interviews, are sampling methods, meaning that the players will be responding at a specific moment in time. If they fill out questionnaires during the game, then we are interrupting the players and modifying their experience of the game. Conversely, if we wait until the end, then they may have forgotten what the real experience was like. Wouldn't it be better if we could capture their experience continuously? Secondly, if we ask players to self-report, such as in interviews or questionnaires, although these are relatively easy to conduct, and can potentially provide a rich source of data, we are relying on their awareness, recall, and cognitive filtering abilities to function before a response emerges, and probably a tainted one at that.

Therefore, self-report methods are far from ideal. In the vast majority of cases, players can't accurately remember their gameplay experience, even after short game sessions. Immediately following gameplay sessions, we often ask players to draw a 'graph' of their experience. In almost all cases, the players draw a line graph, which contains one peak (or trough). In psychology this is known as the serial position effect (Feigenbaum and Simon 1962). Broadly speaking, people tend to remember events at the start, the end, and perhaps one in the middle. Figure 27.1 shows an example of a real player experience diagram that a player has drawn after just a 20-min gameplay session. The player's annotations are anonymized as Player Research were at the time conducting the playtests on this commercially sensitive title.

Although self-report methods are not particularly reliable for understanding the actual player experience, these player experience diagrams do help us address the perception issue: what players think happened, and what they can recall. Interestingly, these diagrams help us identify the gameplay issues that the players may tell their friends about.

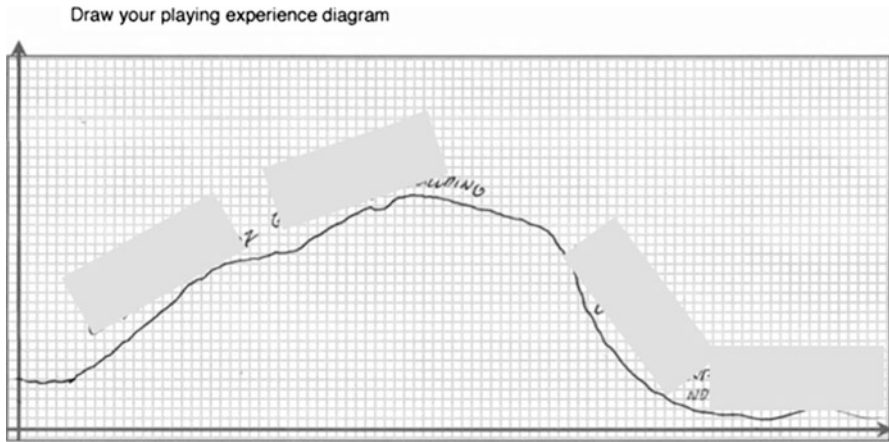


Fig. 27.1 Player experience diagram (anonymized) showing player only recalls events at the start, middle, and end of the 20-min session

There are three reasons for the lack of rich detail in these player experience diagrams:

1. Awareness. Players have to be initially aware that a particular event occurred. This event could be visual, auditory, or cognitive.
2. Recall. As highlighted above, players are quite poor at recalling the subtlety of gameplay. Only the key events are likely to be remembered.
3. Cognitive filtering. Players' true experiences may be tainted by factors, such as trying to please the moderator, excitement at being in a game studio, or playing a pre-release game. What they report is likely to be different than the actual experience. Wouldn't it be useful to capture their experience unconsciously?

So how do we collect this information *continuously* and *unconsciously* from players?

One approach that has been used in user research is facial coding. Paul Ekman⁵ proposes six basic facial expressions that we all have in common, so this seems promising as we don't have to interrupt the player (continuous), and we don't have to ask for their opinion (unconscious). There is also the advantage that these basic expressions are automatic. They occur whether the player likes it or not (Soppitt and McAllister 2011). However, the issue is that in video games, these basic emotions are not usually expressed. In playtests it's common for players to remain quite expressionless, making facial coding very difficult, and not to mention time-consuming.

⁵<http://www.paulekman.com>

Thus, if facial coding, questionnaires, and interviews are not particularly reliable, which other automatic approaches to capturing the player experience can be explored? This is where biometrics shows its potential. This is the subject of the next section.

27.3.2 *Biometrics*

Biometrics, or psychophysiology, is the practice of using sensors attached to the player's body for the purpose of monitoring bodily data. Utilizing biometric data has become an increasingly active area in the video games research community, and several academic studies have been published introducing various biometric-based analysis techniques in video games research. Common physiological measures include Galvanic Skin Response (GSR), facial muscle measures (EMG), cardiac inter-beat intervals (IBIs), and Electroencephalography (EEG). A comprehensive review of the current state of physiological game research, their advantages and limitations has been provided by Kivikangas et al. (2010) and a more recent one by Nacke (2011).

Mandryk (2008) described experiments designed to test the efficiency of physiological measures as evaluators of collaborative entertainment technologies by examining physiological responses to different interactive play environments. Hazlett (2008) describes the use of facial Electromyography as a measure of positive and negative emotional valence during interactive experiences. In addition, Ravaja et al. (2006b) measured facial EMG and cardiac inter-beat intervals in addition to self-report ratings to index physiological arousal and emotional valence. Nacke and Lindley (2009) created a real-time emotional profile (flow and immersion) of gameplay by measuring Electroencephalography (EEG), Heart Rate (HR), EMG, Galvanic Skin Response and using participant eye-tracking. Their results demonstrate correlation between subjective and objective indicators of the gameplay experience (Nacke et al. 2008), showing the potential to provide real-time emotional profiles of gameplay that may be correlated with self-reported subjective descriptions. Other papers in the area, include Yannakakis et al. (2008) who statistically correlated psychophysiological and subjective measures of emotional components of the player experience, Tognetti et al. (2010) who have used physiological data to recognize user enjoyment in a car racing game, and Drachen et al. (2010) who report a case study on GSR and HR correlations with player gameplay experience in a First-Person Shooter (FPS) game.

The use of biometrics does not directly identify the emotion that a player is experiencing. Generally, researchers using biometric approaches may find it difficult to match the obtained quantitative data to the player's emotional experience during play. It is also possible to consider that a player was emotionally aroused not because of specific in-game elements but as a response to an external activity, anticipation, or as a result of something not otherwise observed. The often described 'many-to-one' relationship between psychological processing and physiological response allows for physiological measures to be linked to several psychological structures (Cacioppo et al. 2007).

Fig. 27.2 GSR sensors fitted to the player's hand to capture arousal during gameplay



Some researchers used event-based biometric analysis to construct a player's emotional profile. Nacke et al. (2008) created an automated system that allows reporting of phasic physiological responses at game events. Ravaja et al. (2006a) assess specific game events based on different or contradictory physiological responses triggered by the game events. In this chapter we are following a similar approach by using event-based biometric analysis.

In our approach we do not attempt to map the player's biometrics to a particular emotion, instead we use measures of the players' phasic physiological data purely to log 'micro-events' in the game. In the case of the GSR sensors (see Fig. 27.2), which we use to measure arousal, when we see a peak in the signal, we simply make a note of the timestamp and then replay these events to the player in a post-session interview. These specific moments, identified by peaks in the monitored player's biometric levels, were noted during the playtest, constructing a log of times during gameplay in which a player experienced a potentially meaningful degree of arousal. Micro-events were not analyzed or interpreted at the logging stage, and at no time were individual participant's GSR measurements compared to other players. Instead, after the gameplay session, the gameplay video footage related to every logged micro-event was played back to players, who were asked to relive these specific moments and inform the experimenter of their thoughts. We've found that using biometrics to drive our post-interviews results in more meaningful insights into players' motivations and expectations. A complete explanation of this approach, including advantages and limitation of using biometrics (GSR in our case) in game user research was discussed in Mirza-Babaei et al. (2011).

For example, a biometric micro-event during the use of the quad-bike whilst on an off-road section of the track was logged. In post-playtest interview, the player was asked "Can you explain what happened here?", and the player's response was noted and analyzed. In this example the player responded: "I crashed when I tried to go off-road, I was not expecting this to happen since I have an off-road quad-bike", indicating an issue concerning the player's expectation of the game world. All logged micro-events were addressed in this manner, with usability and player experience issues determined by the players' interpretation of their biological response.

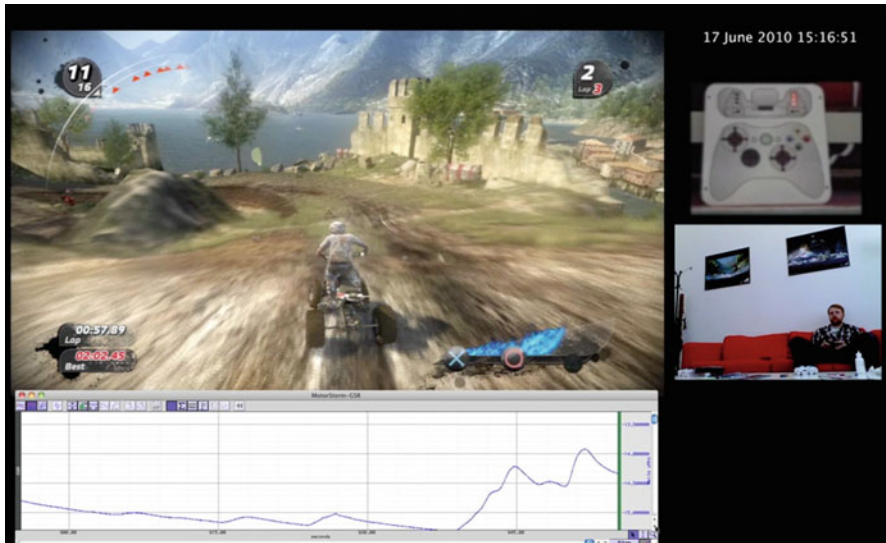


Fig. 27.3 Example image from the gameplay capture setup (PURE images courtesy of Disney Enterprises, Inc.)

During playtests, we can view the game, the players' biometrics (GSR in our case), which buttons they are pushing, and their non-verbal behaviors, all in real time (see Fig. 27.3 showing Player Research's gameplay capture system displaying physiological responses, gameplay, and joypad display). This allows us to observe what the player is doing (behavior) and see the associated experiential reaction at any second.

27.4 Deconstructing Gameplay – Sequential Analysis

In addition to identifying player behavior from the gameplay video, the use of biometrics also shows the corresponding physiological reaction from the player's body. Key gameplay events can be logged, and the player can be asked to explain how they were feeling at a specific moment during post-interview video playback. Thus, the techniques presented in the previous section form a framework for analyzing the coupling between player behavior (what they did) and emotion (how they felt).

Following the game sessions and post-interviews with the player to uncover their feelings at key moments (identified by the GSR arousal peaks), the gameplay videos were analyzed using a sequential analysis method. Behavioral sequential analysis is an approach, which aims to codify and reveal behavioral patterns over time. It's this temporal aspect of the approach that makes it suitable for identifying player patterns, offering insights into gameplay and player profiling.

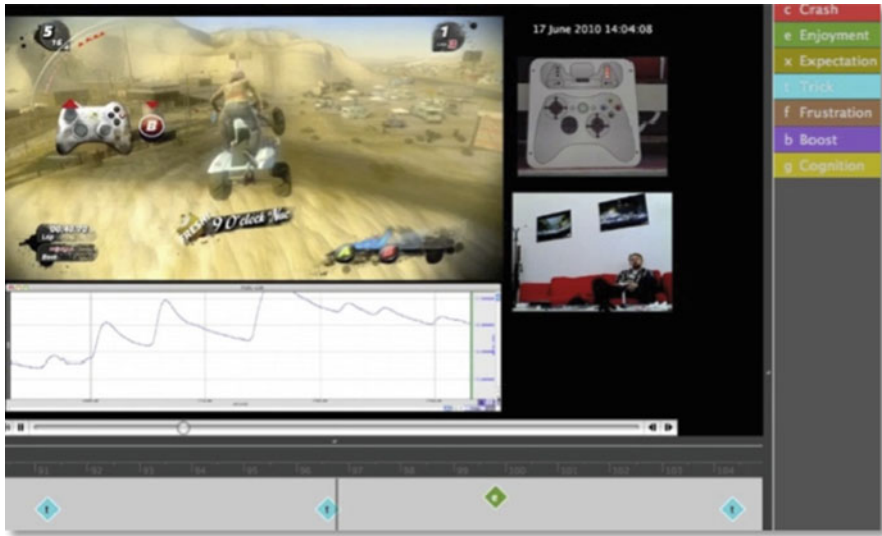


Fig. 27.4 Sequential analysis of gameplay – coupling of behavior (*blue diamonds*) and emotion (*green diamonds*) (PURE images courtesy of Disney Enterprises, Inc.)

Taking a grounded theory approach to initially identify player behaviors, each video was subsequently manually coded for all player actions, e.g., jumping, racing or pulling tricks, as identified from the gameplay video and joypad display. The most time consuming part of this approach is in identifying which codes to use, however once identified, videos can be manually coded relatively quickly (almost in real time). We also coded the associated experience, obtained from post-gameplay interview. For example, as it's not possible to map a peak in the player's GSR signal to a particular emotion, we must ask the player to describe how they were feeling at this particular point and then use that emotion as one of the codes.

Figure 27.4 shows the video annotation application VCode.⁶ The game themes identified by the grounded theory approach are located along the right-hand side, and the individual occurrences of each behavior or reaction are shown along the bottom in a corresponding color of diamond. Although Fig. 27.4 is a static screenshot, in usage the bottom region (sequential coding) scrolls in real-time in sync with the gameplay video. This means it's possible to watch the game and the associated player behaviors and experience in real-time.

As a result of performing the behavioral sequential analysis, the coupling between the player's behavior and resulting experience (if any) is easily visible as a temporal pattern. However, due to the limited visible area of a screen, it's difficult to get a visual overview of the complete player experience. To overcome this issue, we devised a technique, which allows the entire player experience to be scanned quickly. We call these Biometric Storyboards.

⁶ VCode from: <http://social.cs.uiuc.edu/projects/vcode.html>

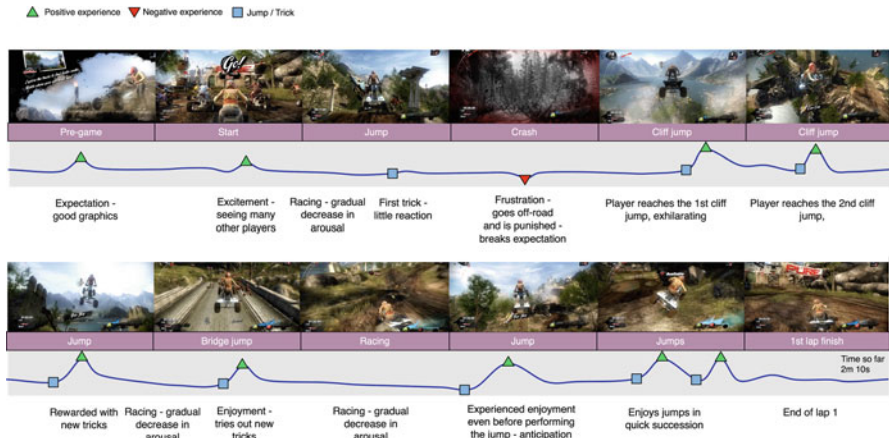


Fig. 27.5 Pure Biometric Storyboard (level 1) (PURE images courtesy of Disney Enterprises, Inc.)

27.5 Visualizing Gameplay – Biometric Storyboards

So far we have shown how our lightweight biometric approach has been useful for identifying moments of gameplay that are of particular interest to the player. However, they’re also particularly useful for improving an often undiscussed area of user research: reporting. Using the correct user research methods and analyzing data thoroughly doesn’t matter one iota if the final results are not communicated well, are not believable, or are not acted on. One approach we developed is a visualization that combines game metrics and player metrics, we called these Biometric Storyboards (Mirza-Babaei and McAllister 2011). The Biometric Storyboard technique is flexible in that it can be used to represent a small section of gameplay with more detail, or longer periods of gameplay in lesser detail.

A game is typically a narrative journey unfolding in time, and our way of effectively communicating the experience is to visualize the player’s complete journey (see Fig. 27.5). The storyboard itself is drawn based on (1) players’ biometric responses (GSR in this case) to log micro-events for post analysis, (2) post-session interviews to explain ‘why’ the changes in their GSR occurred, (3) players’ self-drawn diagrams of their gameplay experience and (4) Sequential analysis of players’ gameplay behavior.

Visualization is a continuously growing area, with research efforts expanding into many different domains, as discussed in Chaps. 18 and 19 of this book. Visualization tools address the challenge of analyzing and presenting overwhelming amounts of data by combining methods from various disciplines, including information visualization, human-computer interaction and data analysis techniques from statistics, data mining, and others.

On the other hand, narrative has always been part of the user experience process communicating how and why a design would work. Storyboards have become

popular techniques for visualizing human-product interaction, not only in design education, but also in design practice (Quesenbery and Brooks 2010). They can help the design team focus on the user's actions and experience. Narrative format ranges from very sketchy to highly detailed, depending on whether they are used to explore new ideas, report existing situations, or present design concepts for criticism and discussion.

Based on feedback from game developers, we're now on our third iteration of Biometric Storyboards. The first version laid out events as a narrative over time. However, as we subsequently learned, time is not always meaningful for some games, and beats (or thematic areas) were considered more representative for version two. The current iteration, version three (see Fig. 27.5), is easier to read again as it couples behavior (the text along the bottom) with the associated player experience (the line graph). As shown in Fig. 27.5, the Biometric Storyboards consist of screenshots depicting key events in the game (mostly identified from players' GSR responses). For this particular game, *Pure*, we have used the upward pointing triangle to denote a positive experience, a downward pointing triangle for a negative experience, and a square to denote tricks and jumps (behaviors). This style of visualization makes it quick and easy to scan the entire player experience and identify individual player patterns between behavior and corresponding emotion.

Biometric Storyboards have been used as a tool to enable discussion. They are easily understandable and use neutral language so that programmers, designers, artists, and producers can all quickly pinpoint areas of the game that are working and those that need refining (Mirza-Babaei et al. 2012). Comparing the Biometric Storyboard version of gameplay with what the player actually remembers (player experience graphs as shown in Fig. 27.1) shows the advantage of using a tool like Biometric Storyboards. The Biometric Storyboard is as close as we have come to representing the true gameplay that a player experiences. Next, we'll put all the previous methods into practice to show how it reveals additional insights into the gameplay experience over traditional user research methods.

27.6 Case Study – Part 2 – Playtesting *Pure*

To evaluate if *Pure* was being experienced as the designer had intended, we employed the previously described biometric approach to identify the exact moments in the game that caused arousal (excitement or frustration). In the following examples, we'll present the findings from one particular player who had no previous experience of *Pure*, but was a fan of the racing game genre. We invited him to play the tutorial, then the first three levels of the game.

On the first track (see Fig. 27.5 which shows the first lap), the player was experiencing arousal moments before the first level began. In post-interview, he said that he was anticipating good things from the game, as the graphics were good and he could see there were many other racers on the track. During the initial moments of gameplay, however, his arousal levels began to decrease. When we analyzed his

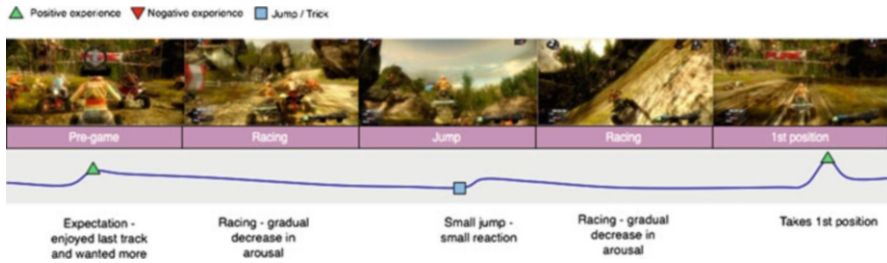


Fig. 27.6 *Pure* Biometric Storyboard (level 2) (PURE images courtesy of Disney Enterprises, Inc.)

behavior, he was merely racing with the competitors, there was no opportunity to do tricks and jumps. However, when he reached the first large jump off of a cliff, there was a sizeable peak in his arousal levels.

This probably comes as no surprise, but what the biometrics can reveal is how often the player experiences arousal. Was this due to the novelty of the first time, or does the player continually enjoy the jumps? This sort of information would be very difficult to determine using traditional user research approaches. What was surprising, however, was that for a racing game, the racing itself did not lead to the most enjoyment. Rather, it was the tricks and jumps. Of course, this was only for this particular player; others may have had a different experience.

For the second level (Fig. 27.6 shows the first lap), as the countdown began to the race we could again identify player arousal, in this case the expectation of racing. The player enjoyed the first level and was looking forward to this one. This didn't happen, however. The level was designed as a quick race, so there were no opportunities for jumps and tricks – the very gameplay elements that had led to fun in the first level. In post-interview, he also mentioned that graphically this was similar to the first track, he was really hoping for a new environment. He did experience arousal, however, when taking first place. Of course, this could be intentional by game designers.

At the start of level three, we were concerned that we wouldn't see any peaks before the race. In other words, he'd lost interest. He did peak, however. In post-interview he said he could tell from the graphics that new experiences were going to be offered, and this intrigued him. The level also provides opportunities for jumps and tricks, and the familiar pattern of jump/trick/enjoyment is constantly experienced throughout most of this level.

However, the player does stop experiencing arousal during jumps for a brief period. At one point, he crashes and falls towards the back of the pack of racers. He immediately changes his gameplay behavior so that he is focused on winning, and only experiences arousal from overtaking other players. Thus, although he was performing jumps and tricks whilst trying to regain first position, he did not experience arousal. This is probably due to his high level of concentration on gaining first position. This change in behavior also reveals interesting information about this player; he is probably more driven by winning in games, then simply having fun Fig. 27.7.

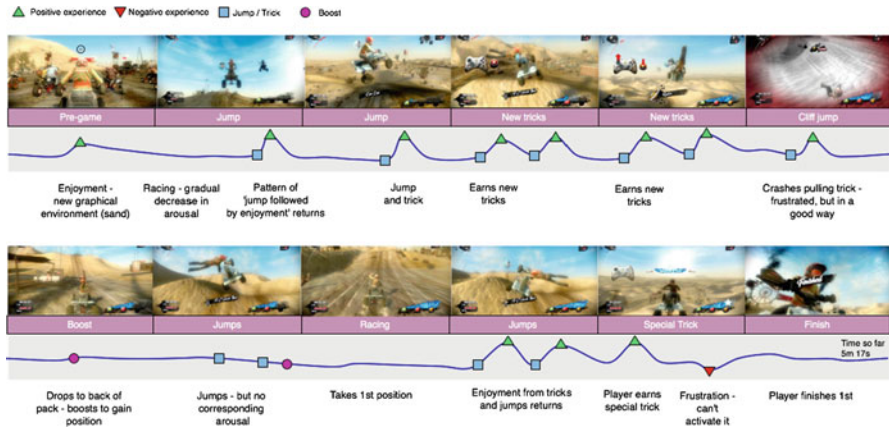


Fig. 27.7 Pure Biometric Storyboard (level 3) (PURE images courtesy of Disney Enterprises, Inc.)

In the case of Pure, biometrics helped us to identify precise moments in which the player experienced arousal. Sometimes this arousal was good, such as when jumping off of a cliff, and sometimes it was bad, such as frustration at confusing gameplay elements. What’s also incredibly useful is that using this biometric approach helps us to identify what Jesse Schell⁷ (2008) refers to as “psychographics.” Whereas demographics identify external factors, such as age, gender, games played, etc., psychographics identify our internal factors that motivate us. As the biometrics can identify the precise moment of gameplay that led to arousal, we can analyze that gameplay event to determine what caused it. Using this approach, we have identified precisely, which game mechanics players enjoy (collecting objects, fighting, exploring) without ever having to ask them during gameplay. Of course, this would all be verified in post-interview, but we have already formed a reasonable profile of who this player is before that stage.

On our player database we also have notes on the player’s psychographics. We find this can be a very useful knowledge tool for improved player recruitment.

27.7 Conclusions

Results of the sequential analysis for this particular player revealed that the fun in Pure does not in fact come from the racing, but from the game mechanics, which appear to be secondary to the racing (jumps and tricks). This may vary by player, but can provide useful feedback to developers, as they can have concrete and specific data on where their game is being enjoyed (or not).

⁷<http://jesseschell.com/>

Some players were observed to change their gameplay behaviors at specific points during the game, confirming that players do not exhibit only one unique game style (e.g. killers or socializers), but are a combination of many, which they flip between depending on game state. This has been noted in the interview with the Pure developer. In particular, the method presented here helps to identify when these key pivotal changes in play styles occur. This can help improve player understanding by using psychographics instead of the more general (and more often used) demographics.

One of the key advantages of this approach is that the output is visual. This provides useful feedback for the developers as they can quickly scan for key differences in level design, player performance and player emotions. Providing easy to interpret feedback to developers is a strong advantage of this user research method.

Although we use biometrics to identify gameplay issues, understand player motivations, and report the player experience, they are not a perfect solution (they are highly subjective measures depending on several uncontrolled factors like caffeine, noisy signals, habituation, etc.). No single user research method is, for that matter. We still have to ask the player to identify which emotion they were experiencing, and they could still lie. The difference is that we have another data point with which to form an opinion. Isn't that where our knowledge and experience as user researchers comes into play? The way we see biometrics is that they help us reduce the amount of uncertainty in explaining an issue, you could say that they add confidence to the findings in our client reports.

If we've analyzed the player's gameplay behavior, seen which joypad buttons they were pushing, noted their various biometric sensor measurements, seen where their eyes were looking, captured their facial expressions, and have access to game telemetry data, then we have a reasonable set of data points to make an informed decision about the player experience. We may also confirm this with the players through traditional interviews and questionnaires.

Understanding the player's interaction with a game takes us on a user research journey covering behavior, rationale, perception and emotion. Video games are highly complex, but are still not as complex as the player. User research methods need to evolve, and the ones outlined here embrace what we believe to be the essential tenets for video game user research, unconscious and continuous. By using these approaches we can get closer to what we're really trying to achieve – making games better.

About the Authors

Graham McAllister is the Director of Player Research, a user research and play-testing studio which provides insights into players and gameplay. Before founding Player Research, he was an academic in the area of Human-Computer Interaction, with interests in designing practical video game user research methods. Graham also writes the column on user research for Edge Online.

Pejman Mirza-Babaei is a Ph.D. candidate at the University of Sussex (UK). His research focuses on developing mixed-methods for a better understanding of user experience in engaging entertainment systems. In particular, he is interested in using physiological measurements in combination with other Human-Computer Interaction (HCI) methods to evaluate the user experience of underdevelopment titles.

After doing a degree in Civil Engineering and starting a short career in engineering consultancy, **Jason Avent** joined Microprose as a Level Designer in 1996. Since then, he has worked at Entertainment Online, ATD, Electronic Arts, Climax (Solent and Brighton) and now Disney's Black Rock Studio. He has also worked in design and production and is currently a Game Director at Black Rock Studio in Brighton.

Acknowledgments Pure images courtesy of Disney Enterprises, Inc.

Selected Bibliography

- Ambinder, M. (2011). Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. Presented in *GDC Vault*.
- Cacioppo, J. T., Tassinary, L. G., & Berntson, G. G. (2007). *Handbook of psychophysiology*. Cambridge: Cambridge University Press.
- Pure* [Xbox 360], Black Rock Studios. Published by Disney Interactive Studios (2008, September).
- Drachen, A., & Canossa, A. (2009). Towards gameplay analysis via gameplay metrics. In *MindTrek '09: Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era*. ACM
- Drachen, A., Nacke, L. E., Yannakakis, G., & Pedersen, A. L. (2010). Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Sandbox 2010*. Los Angeles: ACM.
- Feigenbaum, E. A., & Simon, H. A. (1962). A theory of the serial position effect. *British Journal of Psychology*, 53, 307–320.
- Hazlett, R. (2008). Using biometric measurement to help develop emotionally compelling games. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advancing the player experience* (pp. 187–205). San Francisco: Morgan Kaufmann.
- Kivikangas, J. M., Ekman, I., Chanel, G., Järvelä, S., Salminen, M., Cowley, B., Henttonen, P., & Ravaja, N. (2010). Review on psychophysiological methods in game research. In *Proceedings of 1st Nordic DiGRA, DiGRA*
- Mandryk, R. (2008). Physiological measures for game evaluation. In K. Isbister & N. Schaffer (Eds.), *Game usability: Advancing the player experience* (pp. 207–235). San Francisco: Morgan Kaufmann.
- Mandryk, R. L., & Atkins, M. S. (2007). A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human Computer Studies*, 65, 329–347.
- Mandryk, R. L., Atkins, M. S., & Inkpen, K. M. (2006). A continuous and objective evaluation of emotional experience with interactive play environments. In *Proceedings of the conference on human factors in computing systems (CHI 2006)*. Montreal: ACM.
- Mirza-Babaei, P., & McAllister, G. (2011). Biometric storyboards: Visualising meaningful gameplay events. In *CHI 2011 BBI workshop proceedings*, Vancouver, BC, Canada.
- Mirza-Babaei, P., Long, S., Foley, E., & McAllister, G. (2011). Understanding the contribution of biometrics to games user research. Full research paper at *Think Design Play: The fifth international conference of the Digital Research Association (DiGRA 2011)*, Hilversum, The Netherlands.

- Mirza-Babaei, P., Nacke, L., Fitzpatrick, G., White, G. R., McAllister, G., Collins, N. (2012). Biometric storyboards: Visualising game user research data. In *Proceedings of CHI EA 2012*, Austin, TX, USA.
- Nacke, L. E. (2011). Directions in physiological game evaluation and interaction. In *CHI 2011 BBI workshop proceedings*, Vancouver, BC, Canada.
- Nacke, L., & Lindley, C. (2009). Affective ludology, flow and immersion in a first-person shooter: Measurement of player experience. *Loading* 3(5). <http://journals.sfu.ca/loading/index.php/loading/issue/view/6>.
- Nacke, L., Lindley, C., & Stellmach, S. (2008). Log who's playing: Psychophysiological game analysis made easy through event logging. In *Proceedings of fun and games, second international conference* (pp. 150–157). Eindhoven: Springer.
- Nacke, L., Grimshaw, M. N., & Lindley, C. A. (2010). More than a feeling: Measurement of sonic user experience and psychophysiology in a first-person shooter game. *Interacting with Computers*, 22(5), 336–343.
- Quesenberry, W., & Brooks K. (2010). *Storytelling for user experience*. Edited by Marta Justak. New York: Louis Rosenfeld.
- Ravaja, N., Saari, T., Salminen, M., Laarni, J., & Kallinen, K. (2006a). Phasic emotional reactions to video game events: A psychophysiological investigation. *Media Psychology*, 8(4), 343–367.
- Ravaja, N., Timo, S., Marko, T., Jari, L., Mikko, S., & Matias, K. (2006b). Spatial presence and emotions during video game playing: Does it matter with whom you play? *Presence: Teleoperators and Virtual Environments*, 15(4), 381–392.
- Schell, J. (2008). *The art of game design*. Amsterdam: Elsevier/Morgan Kaufmann.
- Soppitt, M., & McAllister, G. (2011). Understanding player experience using sequential analysis. In *Think design play: The fifth international conference of the Digital Research Association (DIGRA)*, Hilversum, The Netherlands.
- Tognetti, S., Garbarino, M., Bonanno, A. T., Matteucci, M., & Bonarini, A. (2010). Enjoyment recognition from physiological data in a car racing game. In *Proceedings of the 3rd international workshop on affective interaction in natural environments (AFFINE '10)* (pp. 3–8). New York: ACM.
- Yannakakis, G. N., Hallam, J., & Lund, H. H. (2008). Entertainment capture through heart rate activity in physical interactive playground. *User Modelling and User-Adapted Interaction*, 18(1), 207–243.

Part VI

Analytics and Player Communities

This part is concerned with understanding player behavior in Massively Multiplayer Online Games, where the social dimension adds a completely new depth to the actions available. The contributions explore how the social architecture in such games are designed to foster collaboration and maximize opportunities for player-to-player interactions.

The take-aways for this part focus on analysis of game metrics using different hermeneutic grids of:

- Psychology of personality and motivation
- Sociology
- Economic theories

The part consists of three chapters:

- Chapter 28: *Data Collection in Massively Multiplayer Online Games* discusses the use of game metrics to model a wide range of phenomena, such as the social dynamics of online groups or the relationship between a player's personality and their behavior. This is a contribution by Nic Ducheneaut, senior scientist, and Nick Yee, research scientist at PARC.
- Chapter 29: *Designer, Analyst, Tinker: How Game Analytics Will Contribute to Science* discusses the use of game telemetry to analyze and model player behavior and the relevance of such research to science. This contribution is by Edward Castronova, Travis L. Ross and Issac Knowles, faculty from Indiana University.
- Chapter 30: *Interview with Ola Holmdahl and Ivan Garde from Junebud* is an interview with Ola Holmdahl, founder of Junebud and CEO and Ivan Garde, producer, business and metrics analyst. This interview explores the use of metrics for an MMO developed by Junebud.

Chapter 28

Data Collection in Massively Multiplayer Online Games: Methods, Analytic Obstacles, and Case Studies

Nicolas Ducheneaut and Nick Yee

Take Away Points:

- MMOGs are large-scale, persistent social engineering experiments that can generate a goldmine of behavioral data.
- This data can be used to model a wide range of phenomena, such as the social dynamics of online groups or the relationship between a player's personality and their behavior.
- In turn, these models can be used to improve current and future games by making them more responsive to their players' needs and motivations.
- The richness of data in MMOGs can pose significant analytical issues, such as the "garbage in, garbage out" problem: we suggest collection and normalization strategies to generate useful variables.

28.1 Introduction

28.1.1 For Fun and for Profit: Why Collect Data from MMOGs?

Massively Multiplayer Online Games (MMOGs) occupy a unique position in the videogaming landscape. While multiplayer computer games are certainly not new (Spacewars, in 1962, was already designed for two players), MMOGs have gone much farther than any other genre in their attempts to encourage social interactions between large groups of players. This is all the more interesting when considering

N. Ducheneaut • N. Yee (✉)

Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

e-mail: contact@nickyee.com

that most of the activities offered in a MMOG, such as killing monsters, leveling up a character, gaining more powerful abilities and gear, etc., are all already available in single-player games: from an individual standpoint, one could have a strikingly similar experience playing a MMOG like *World of Warcraft* (Blizzard, 2004), for instance, and a single-player role-playing game (RPG) like *Dragon Age* (Electronic Arts, 2009). Therefore, what makes MMOGs so unique and interesting is their social architecture: the way each game world is designed to maximize opportunities for player-to-player interactions. In turn, it is these repeated player interactions that keep subscribers coming back – and paying their monthly fee: as previous studies have shown “it is the players that are addictive [in MMOGs], not the game itself (Lazarro 2004).” Understanding the nature and structure of player behaviors and interactions in online games is therefore not only an interesting (and fun) sociological question: it is also potentially a very lucrative one, since a better understanding of player behavior could lead to improved social architectures with even higher attractiveness to the players and a corresponding increased retention (and profit).

The basic template for MMOGs has remained broadly stable for the past decade. The massive popularity of *World of Warcraft* (WoW), which was based on principles borrowed and refined from older, pioneering MMOGs like *Everquest* and *Ultima*, has a lot to do with this stability: the game has been available for 7 years and remains in a class of its own, with more than ten million subscribers by most accounts (see for instance: <http://mmodata.net>). Clearly, such longevity indicates that some useful lessons could be learned from observing behaviors in WoW, perhaps leading to a better understanding of the key elements of today’s “standard” social architecture for online games. Furthermore, it might also help future games break the current mold and offer some (perhaps overdue) novelty by addressing design issues that might not be immediately apparent to players and designers alike. Finally, observing and analyzing data from a large, popular MMOG like WoW may also suggest ways of better instrumenting future games and designing applications to leverage the data in productive ways for game designers and producers (for instance, by monitoring the social life of groups and making sure that the desired interactivity is achieved). Before looking at any data, it is useful to describe the main components of the formula used by WoW and others to achieve “collaboration by design” in their social architecture. This is the subject of the next section.

28.1.2 Collaboration by Design: The Current MMOG Template

The current crop of “A-list” MMOGs (that is, those having reached 100,000+ subscribers like WoW, Rift, etc.) relies on a simple formula that is almost identical across all games, with some minor variations. The archetypal progression goes like this: players start as level 1 characters in a vast, unfriendly world that they share with the other players. Their character then gains levels and more powerful abilities through quests or missions. But while these quests are reasonably straightforward to complete alone for the first few levels, they progressively become difficult enough

that players need a group to succeed and progress. Initially, they can rely on whoever is around at the time and form a pick-up group to accomplish their goals. But as they get closer to the highest possible character level, group activities become complex enough that they cannot rely on pick-up groups to succeed anymore. The endgame “instances” (dungeons populated by very tough monsters) in WoW, for instance, can require groups of up to 40 players to join forces for encounters lasting several hours. Anyone who tries recruiting 39 random people to join them for the next 6 h in a dungeon will quickly realize this is close to impossible, unless they already have some pre-existing relationship with the people they are trying to recruit. In MMOGs, this repeated need for increasingly complex collaboration leads to the creation of stable, long-term player associations: the guilds, which are used as a stable and reliable pool of teammates for large-scale social activities.

The collaboration template above is reinforced in turn by another game mechanic: classes. Indeed, users can create characters with different skill sets that complement each other. For example, heavily-armored tank classes shield the group from enemy attacks, while lightly-armored damage dealing DPS (damage per second) classes deal damage to enemies and healing classes restore health lost in combat. A successful group (and guild) will have to recruit players with complementary skills to function properly.

The above notwithstanding, it is worth noting that players do not spend all their time in groups. To keep them interested, “modern” MMOGs like WoW offer a very broad range of activities. People who never play these games often assume they are mostly about combat, but there are many other pursuits players can engage in, from collecting cute companion pets to calmly fishing in a remote corner of the world. In turn, a system of *achievements* keeps track of both combat and non-combat based objectives. In WoW, there are Achievements for the zones explored, dungeons completed, number of hugs given, and cooking proficiency, to mention a few examples. These *achievement* scores provide a good sense of how a player chooses to spend their time in WoW, be it inside or outside of groups. In fact, this brings us to another key feature of most MMOGs: *their instrumentation*.

28.1.3 A Metrics Goldmine

MMOGs, like WoW, offer a wide and varied set of rich behavioral cues to draw from. From class choice to amount of player-vs.-player (PvP) activity, from number of emotes used to amount of world exploration, the game context offers a range of measurable behaviors. The client-server architecture of the game’s software makes tracking these behaviors relatively straightforward: in the end, any player action translates into a database update, which can be logged and mined over time for interesting patterns. To be sure, some amount of forethought makes data analysis in MMOGs more straightforward: database tables can be structured to make querying information about key behaviors easier, for instance. But broadly speaking, the persistent nature of the game world and the corresponding software required to support it, make each MMOG a behavioral metrics goldmine.

Until recently however, most of these metrics were hidden from public's view and saved for internal use at game companies. Therefore, tracking player behaviors "from the outside" required some ingenuity but was not altogether impossible (we illustrate this in our next section). Blizzard, the developer of WoW, is unique in that they decided in 2007 to offer public access to much of their internally-collected data at a website known as the Armory. In short, by searching for a character's name, anyone can view details about their past activities, including how many hugs they have given, the quality of their equipment, the class they prefer to play, etc. More importantly, these metrics have been tracked since the character was first created. With a few clicks, we can gather a character profile that has cumulative data over many months of game play. It bears emphasizing the tremendous social science research opportunities that are made possible by this publicly-available database of longitudinal behavioral metrics. Based on this data, we can illustrate the range of data mining and analyses that can be done by game companies to leverage insights from the social sciences in order to optimize their games' social architecture. Before doing so however, it is useful to spend some time discussing the evolution of our data collection methods to illustrate what can be done both when limited and (nearly) limitless data is available from a game, as well as some complementary sources of data that are often needed.

28.1.4 Collecting Data in MMOGs

Our team at PARC has been building a kind of virtual observatory to take advantage of the opportunity for social science research offered by MMOGs since 2004. We have looked at many games, including Everquest (Sony Online Entertainment, 1999), Star Wars Galaxies (Sony Online Entertainment, 2003), and City of Heroes (NCSOFT, 2004) (Ducheneaut and Moore 2004, 2005; Moore et al. 2007). For this chapter, however, we will be focusing on our largest effort to date, that is, the work we have been doing in World of Warcraft.

Our approach to collecting data in WoW evolved over the years as Blizzard progressively decided to release more data for public consumption. Shortly after the game's launch, no public source of player behavioral data was available. However, Blizzard designed WoW to be extensible through the use of "addons": small programs written in Lua, a scripting language, that could be designed by players to extend or refine the game's user interface (for instance, displaying key information such as the health of group members during a raid in a more visible and accessible way). An in-game API (application programming interface) lets each Lua script access a limited set of game functions, in order to guarantee that addons cannot be used to give some players an unfair advantage (for instance, by automating their combat activities). Interestingly for us, one of these API functions could be used to gather limited, but valuable data about the players: the "/who" command.

For a player, typing "/who" in-game lists all the other players in the same game zone who are roughly the same level (plus or minus 5 levels), with an upper limit of

49 results returned. The intent is to facilitate the formation of groups by quickly finding compatible groupmates. However, the command can be expanded to include additional parameters, such as specifying a different game zone, or a different level range. Therefore it becomes possible to conduct a census of the entire population on a given server at a given time by progressively cycling through small segments of the population (e.g., only players of a given class and level in a given zone), aggregating small batches of 49 players or less over time to exhaustively cover all players.

We designed a WoW addon to perform such a census. Based on rate limits for each “/who” query and depending on server load, our addon captures a list of all active players on a server every 5–15 min. Each time a character is observed our software stores an entry of the form:

```
Alpha, 2005/03/24, Crandall, 56, Ni, id, y, Felwood,
Ant Killers.
```

The above represents a level 56 Night Elf Druid on the server Alpha, currently in the Felwood zone, grouped (“y”), and part of the Ant Killers guild. Using this application, we collected data continuously from June 2005 to December 2006 on five different servers: PvE(High) and PvE(Low), respectively high- and low-load player-versus-environment servers; PvP(High) and PvP(Low), their player-versus-player equivalents; and finally RP, a role-playing server. Overall, we observed roughly 300,000 unique characters. Through that data it became possible to see where players are in the game world and whom they are playing with, which can then be used to reconstruct the social networks formed in guilds and elsewhere. As we will illustrate in our “data highlights” section, even a limited number of variables such as the above can still be leveraged to model some key aspects of player behavior: the lack of well-formatted and easily accessible data is therefore not necessarily an impediment to conducting research to better understand how MMOGs function and how they can be improved.

As we mentioned earlier, Blizzard decided in 2007 to make more of their internal data publicly accessible, in the form of the WoW Armory. The site evolved from a relatively concise summary of character statistics (e.g. level, race, class, equipment, basic combat statistics) presented through XML pages to a full-fledged data API that can be queried using well-documented HTTP requests and provides access to almost all aspects of a character’s activities. For each active character, more than 3,500 raw behavioral metrics are available via the Armory API. We, therefore, were able to develop an increasingly complex and robust set of Web scrapers to collect all the information available about a character in our sample on a daily basis (the Armory is updated once a day). The corresponding range of data most closely reproduces what producers and managers would have access to when monitoring the evolution of their game.

Finally, we also have access to survey data from volunteer players. This tells us about the players’ demographics: information like age, gender, education, etc. We can then see how these demographics relate to their in-game behavior by looking at their characters’ activities in the Armory and the data collected by the add-ons. Such “real life” information is crucial when trying to make sense of the in-game

behavioral data, since a player's profile often influences their choice of activities and how they behave online (which we will illustrate in more detail shortly). We currently have a sample of 500 active players in the US and another 500 from Hong-Kong and Taiwan participating in our studies. A new phase of our project started in September 2011 and we are currently recruiting players from Mainland China and Europe too.

28.1.5 Data Analytic Obstacles

In retrospect, our first batch of data from the “/who” addon was simple in the sense that while we collected data from hundreds of thousands of characters, only seven variables were gathered per character. Furthermore, the unit of analysis was the same as the unit of data collection (i.e., character), since we did not have access to actual player data at the time (we began recruiting volunteers in 2009). Thus, the only minor data analytic obstacle was the processing time it took to traverse the large character set. In comparison, the more advanced Armory scrapers let us gather over 3,500 variables per character, and more importantly, the unit of analysis (i.e., player) was different from the unit of data collection (i.e., character). In this section, we will present some of the analytic challenges we encountered with this more complex data set and some of our solutions. Indeed, a key issue in making sense of activities in MMOGs is the familiar “garbage in, garbage out” problem: the analyses will only be as good as the data that is used to make them. As we show below, most of the “raw” data available from MMOGs needs to be carefully pre-processed to reach meaningful results.

28.1.5.1 Verbosity and Granularity

With the data collection architecture for the WoW Armory in place, we had assumed, erroneously, that the accumulated XML data would be relatively clean and easy to parse. After all, these variables were being stored in a structured database format in Blizzard's back-end, and all the variables were uniquely named and sorted in meaningful categories (e.g., combat, social). Unfortunately, the Armory variables were incredibly verbose and sometimes unintuitive. As one example, defense is a game mechanic that determines damage avoidance in combat. In the Armory XML, there are 5 variables related to the total defense value, labeled as: decreasePercent, increasePercent, plusDefense, rating, and value. In our initial pass over the data, we had assumed that “value” was the most relevant variable to extract, but it turned out that “value” was a static variable of 400 across all characters. Extracting meaningful variables for analysis thus required much more manual crawling and cross-checking of the Armory variables than we had assumed.

The scale and granularity of the Armory data posed a different obstacle. Much of the Armory data is presented in nested hierarchies of increasing granularity. For example, there is a Boolean variable of whether each geographic zone has been

visited (over 250 zones). These are then aggregated at the region level, and then continent level. As social scientists who care about understanding virtual communities, we realized that extracting the fine-grained variables would not be conceptually meaningful. For example, it would be hard to interpret what having visited any one particular zone meant. On the other hand, a high-level ratio of all zones visited (i.e., zones visited/total zones) would more readily map to a meaningful concept of geographic exploration. Thus, we attempted, where possible, to generate aggregated variables at a level where they would be easier to interpret.

28.1.5.2 Aggregation and Normalization

The most significant analytic challenge was reconciling the unit of analysis with the unit of data collection. While we collected data for individual characters, our goal was to understand how players behaved in the virtual world. This mapping is complicated for several reasons. First of all, players can, and often do, have more than one character. Second, different players have different numbers of characters. Third, character metrics are greatly influenced by character level, especially at the expansion break points—to provide a level playing field for each expansion, Blizzard greatly increases equipment and skill powers at these break points. Fourth, due partly to the expansion break points, character metrics do not scale linearly with character level. And finally, character metrics are also greatly influenced by time played. As much as possible, we would like to be able to examine in-game preferences isolated from playing time itself.

Thus, even though we knew that character-level metrics had to be aggregated at the player level, it wasn't at first clear what aggregation method to use. As a crude example, we knew that simple averaging would not work. Say Player A has one level 80 character, and Player B has one level 80 character and a level 1 character. If we simply calculated the average of character metrics, then Player B would be unfairly penalized for his/her level 1 character, even though intuitively that shouldn't be the case. In short, the problem of aggregation is inextricably tied with the problem of normalization—if we couldn't figure out how to compare a level 1 character's metrics with a level 80 character's metrics, then we would not be able to solve the character aggregation problem either. Nor would we be able to solve this problem even if we only used the "main character" from each player. After all, this still would not solve the normalization issue even if it side-stepped the aggregation issue. And it also seemed like a great waste to discard so much actual data.

After much deliberation, we realized that while normalized aggregation was necessary, it was not necessary to have a single, unified method to derive all the player-level variables. To this end, we adopted a set of 5 variable normalization strategies:

- **Static Character Attributes.** For static attributes such as character gender, we can normalize the variable against the total number of characters. Thus, for example, we can derive a "male character ratio" by taking the number of male characters and dividing this by the total number of characters.

- **Variable Character Attributes.** Some character attributes vary over time – combat role is a good example. In team-based combat, characters are almost always specialized in one of 3 combat roles: healing, tanking, or damage-dealing. Characters can switch their specialization and thus combat role is an interesting variable that changes over time. Such variables can be normalized against time played. Therefore, we can calculate a “tank ratio” by tallying the number of Armory days that any of the player’s characters were in the tank role, and then dividing this number by the total number of Armory days for that player. This “tank ratio” would then provide a level-independent, time-independent, player-level measure of preference for the tank role.
- **Partitioned Variables.** There are also variables that are already partitioned by the Armory, many in nested hierarchies. The achievements are a good example. The Armory tracks the completion of all individual achievements, and then tallies these into achievement categories. Thus, we can calculate an “exploration ratio” by taking the exploration achievement score (across all of a player’s characters) and dividing this by the total achievement score (across all characters). This “exploration ratio” would then give a sense of how much of their playing time, relative to other players, is spent exploring the world.
- **Cannot be Normalized, and Level-Dependent.** There are many raw count metrics in the Armory that cannot be normalized using the above three strategies. Moreover, they are also dependent on character level. The collection of vanity pets is one good example. It is easier for high-level characters to collect vanity pets and there is nothing to really normalize this count against. For these variables, we extracted the maximum count across a player’s characters. We extracted the maximum because we felt it was more relevant that any one character collected a large number of vanity pets, as opposed to five characters each collecting a small number.
- **Cannot be Normalized, and Not Level-Dependent.** Finally, there are raw count metrics that are not tied to a character level. The emote “/hug” is a good example. Any character, regardless of level, can hug and often as they like. In such cases, we calculated the sum of hugs across a player’s characters.

For our eventual analysis of the collected Armory data, we relied on these five strategies to generate variables that were conceptually meaningful. We submit that game companies interested in tracking player behavior could benefit from designing their data infrastructure such that pre-normalized metrics are immediately available for analysis: this way, their analysts could jump straight to making sense of relevant patterns, instead of spending valuable time writing and running data pre-processing applications.

28.2 Data Highlights and Design Implications

With this background in mind, we will now illustrate the kind of research that can be done in MMOGs based on highlights from our studies in World of Warcraft. We will focus on three issues, pointing out how each of them suggests concrete

game design implications for current and future MMOGs. As such, our intent in this section is not to demonstrate the scientific potential of online games (even though such potential clearly exists, see for instance (Castronova 2006)), but rather to show how principled research and data analysis can generate valuable insights for the gaming industry.

The first area is sociability. WoW's designers managed to attract ten times more subscribers than its closest competitor: it is reasonable to hypothesize that they must have done something right with the game's social environment. We will take a look at some data showing what makes WoW a unique social space compared to other online game and see if similar designs could be used elsewhere (Ducheneaut et al. 2006).

The second topic is collaboration. Given that games like WoW are designed explicitly to encourage collaboration, it is interesting to ask what makes a group successful in that environment. Understanding the nature and structure of groups in WoW could potentially help design future games such that collaboration between the players is easier and more enjoyable (Ducheneaut et al. 2007).

Finally, the third topic is player personality. Given the wide array of things one can do in WoW, it makes intuitive sense that the choices player make could say a lot about who they are in the real world – for instance, PvP-combat fanatics might well be aggressive outside the game too. Finding a link between player personality and in-game behavior would be potentially quite useful, since it would enable game designers to adjust a game's content based on a player's profile (Yee et al. 2011).

Before moving forward however, we want to emphasize that this chapter only provides an overview of our results in each area. Interested readers can find the full presentation of our findings in the papers referenced in each paragraph above.

28.2.1 Sociability in World of Warcraft

As we mentioned previously, MMOGs are designed to encourage social interactions. Not all social experiences are equal, however, and there is a widespread notion, dating back to the first widely successful MMOGs like Everquest, that intense social experiences are key to MMOG design. In other words, many believe the design should encourage players to spend long stretches of time collaborating directly with each other in dungeons requiring 5, 10, 20, maybe even 40 guild members to be present for several hours – and indeed that is exactly how the “endgame” is structured in WoW, after players have reached the highest level for their character. If they play long enough, they will eventually find themselves “raiding” a lot since it is the only avenue for progress left to them at this stage (bosses in high-end raids “drop” the best pieces of equipment, which can be used to improve the character's attributes without leveling up). It is true that these raids can be very rewarding experiences. Beyond the loot players get for their character, there is something very satisfying about defeating a high-end boss with 39 well-coordinated teammates (Chen 2009; Taylor 2006).

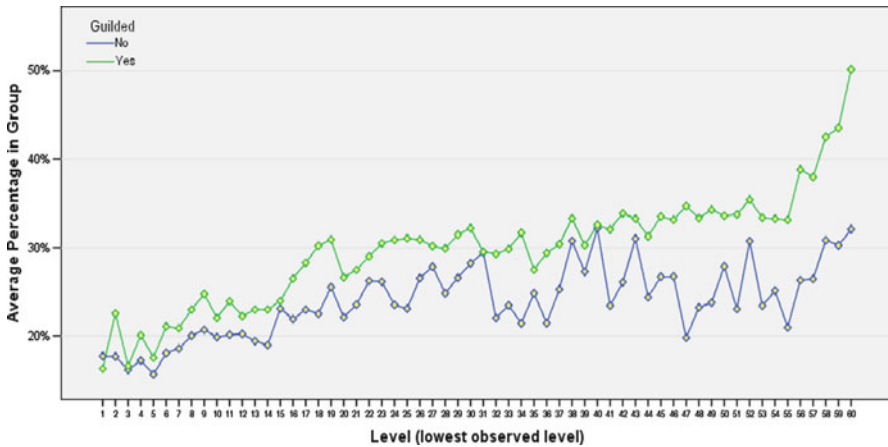


Fig. 28.1 Average percentage of time spent in groups across all levels, for guided and non-guided players

The question, however, is whether or not this is really what most players want when they start playing the game. As we are about to show, we think *WoW*'s success can be explained in part by the way its designers decided to support less intense forms of sociability that are equally valuable. First, and contrary to popular belief, our data shows that *WoW* players spend a large fraction of their online time alone. Figure 28.1 shows the percentage of time players spend in a group based on their character's level. One can see that while it is true that players eventually spend a significant fraction of their time in groups, this happens mostly after they have reached the "endgame", that is, the highest level for their character, and only if they decide to join a guild. Indeed at the endgame their only option to continue playing is to group with others to enter the raids we were describing earlier and win powerful items, which are now the only way to progress since their level is capped. But note on this graph how a significant chunk of a player's early tenure in the game is spent alone, and this even if they decide to join a player association early on.

Therefore, one interesting design component of *WoW* here is how players are socialized into groups very slowly. *WoW* is almost two games in one, an individual leveling game in the early stages and an intensely collaborative group combat game in the later stages. In between the two, players are slowly pushed into more and more group quests, but it is never an obligation and a lot of activities can be done alone. This gives the players time to be absorbed into the game world without feeling that they are "forced" to play with other people. In other words, the first part of the game acts as a "gateway drug" into raiding. A lot of older MMOGs were much more "hardcore" and almost forced players to join a guild right from the beginning. *WoW*'s design probably attracted a much larger subscriber base in part because of its more gradual approach to socialization.

Another interesting design component is how the game offers several character classes that are easier to play alone: for instance, Warlocks can summon powerful demons to assist them, which means that Warlock players are essentially controlling

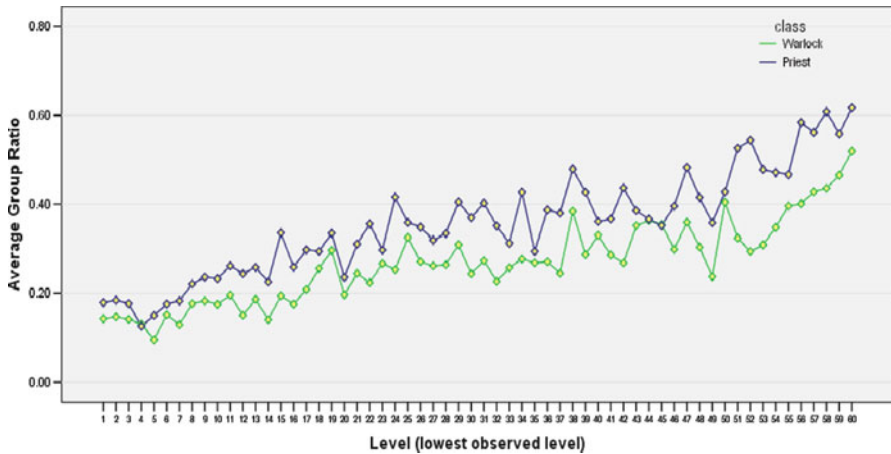


Fig. 28.2 Average percentage of time spent in groups for two classes (Warlock and Priest)

a 2-man group by themselves. This greatly increases their chances of survival. By contrast healing classes like Priests have few offensive abilities and have trouble functioning correctly without support. This is reflected in the time players of each class spend in groups, as can be seen on Fig. 28.2 – players of Priests consistently spend almost 10% more of their time in groups than Warlocks. But interestingly, the “solo” classes are also among the most popular: our data reveals that 15% of the player population plays Warlock, for instance, versus 9% for the Priest. Given the choice, it seems players like having the option to play by themselves if possible and they choose their character classes accordingly. This was probably another good design choice by Blizzard: allowing “solo play” in a multiplayer game.

The consequences of these two design decisions can be seen in the way guild members interact with each other. Again, the popular image of guilds in MMOGs is that of a large group of raiders constantly playing with each other. But given the way WoW is structured, guilds turn out to be quite different. When we re-constructed guild social networks from our data we realized that players interact with few of their guildmates. Many guilds look like Fig. 28.3, with a core of interconnected players grouping constantly with each other and a larger number of more peripheral or even disconnected players. On average, players interact with only 1 out of 4 of their guildmates.

Overall, a key finding from our studies was that, because of the way WoW is designed, players spend a great deal of their time surrounded by other players, not necessarily interacting directly with them (Ducheneaut et al. 2006). We would argue that this is perhaps precisely the reason why WoW has become so successful. Unlike its more “hardcore” predecessors, WoW was one of the first MMOGs designed to support a lot of individual activities. But while these activities are individual, they still take place in a multi-user environment, in the presence of other players. In that context other players still play important roles, even if they are not groupmates during a quest.

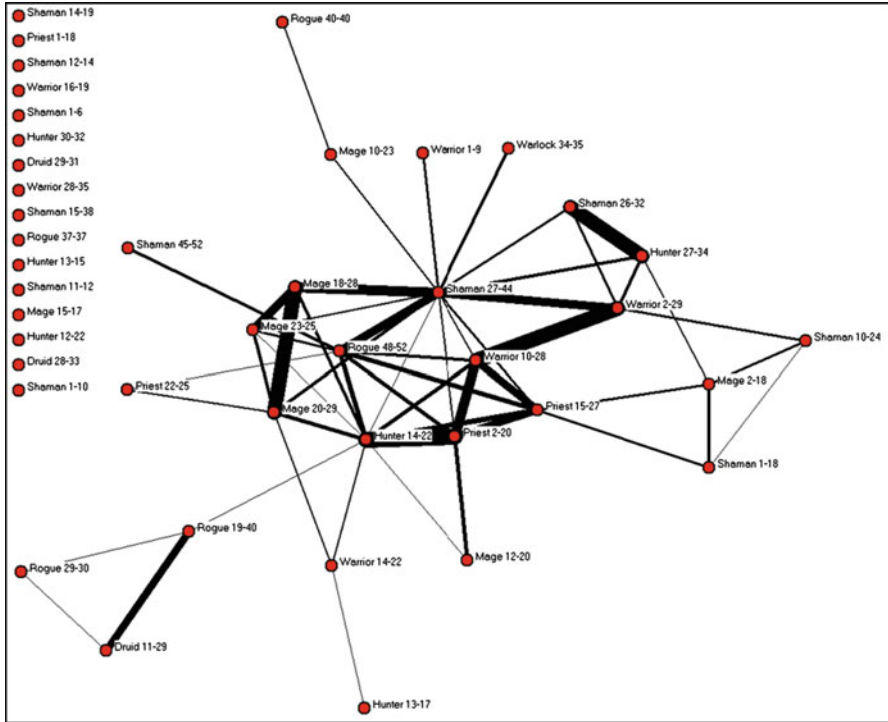


Fig. 28.3 Social network for one illustrative guild

WoW’s designers understand that being surrounded by other players creates a strong sense of social presence in the game. We need to remember that interactions can take place outside of groups and guilds: for instance it is always possible to chat, whenever and wherever a player wants to. In fact, one player said the game is more like “World of Chatcraft” to him: he plays alone, but he constantly exchanges messages with his guildmates along the way. The other players are also a source of constant spectacle: wherever a player is, there are always other players around conducting various activities, quite often humorous ones – for instance, several guilds have organized naked gnomes marathon between two cities in the game world: just seeing the runners go by is entertaining, there is no need to group with anyone to enjoy it. Activities like these make the world feel real, populated by other human beings instead of automatons. As Ted Castronova (2005) has argued, the presence of other human players in these games validates emotions, and we think this is apparent in the ways sociability takes place in WoW. The analogy, here, is that playing WoW is like reading a book or working on a laptop in a crowded café: the activity can be individual, but conducting it in a bustling social space adds a certain comfort to it.

Our data therefore leads to an interesting design lesson: even when creating a group-based, collaborative environment like a MMOG, it still pays to think carefully about what HCI researcher Steve Reeves and his colleagues (Reeves et al. 2005)

have called the “spectator experience” – all the ways that social interactions can be experienced indirectly. If anything, the recent success of so-called “casual” multi-player games on Facebook shows why this matters a great deal: they support exactly that kind of indirect social gaming experience.

Still, when players spend time with each other, guilds frame their social experience and play a central role in creating and maintaining ties between their members. In our next section, we will focus on some of our data that sheds some light on how to support these important player associations.

28.2.2 *The Life and Death of Guilds*

Our data clearly shows that guilds vary greatly in size, longevity, and membership (Ducheneaut et al. 2007). Therefore, it is interesting to look at their structure and see what it takes to create and sustain a successful guild in WoW. An important thing to note here, is that while we are talking about games, the complexity of tasks that guilds have to accomplish to reach high-end dungeons is quite high, up to a point where playing sometimes looks more like working (Yee 2006a). Players cannot simply enter a raid dungeon on a whim, there is a lot of preparation involved: getting the necessary material and gear, scheduling the run with many geographically-distributed players, discussing tactics and group formation, etc. There is also the issue of loot: the bosses in the dungeon will drop only a few of the most coveted items, and guilds have to decide who will get what. On top of this there are unavoidable interpersonal issues, just like in any other group. Because of this complexity, many guilds do not survive very long.

A simple metric from our data that immediately caught our attention was the distribution of guild sizes. The median guild size is nine players, quite small, but more importantly the 90th percentile of the distribution is at 35, with a few very large guilds of more than 100 members forming the long tail of the distribution. Hence, it seems that growing a group in WoW beyond 35 members is quite difficult and out of reach for the large majority of guilds. This is interesting considering that in the early days of the game the most advanced dungeons required 40 players, and were therefore inaccessible to most guilds. Even more interesting, Blizzard has since then changed most high-end instances to 10 or 25 players, below the knee of the distribution and therefore much more accessible.

More generally though, this distribution suggests that there are limits to collective action in online games like WoW. The anthropologist Robin Dunbar (1993) became famous by proposing that there is a cognitive limit to the size of groups that humans can maintain, and his studies pegged this number at around 120. Clearly most guilds in WoW are far from that size: it probably pays to design activities for smaller social structures instead.

Our data also revealed other important properties of guilds in WoW. Guilds are highly unstable, with a monthly churn rate of 25% – that is, each month, a quarter of a guild’s membership leaves, to be either replaced or leading progressively to the group’s disappearance. In fact, the death rate for guilds over 6 months is 54%, so

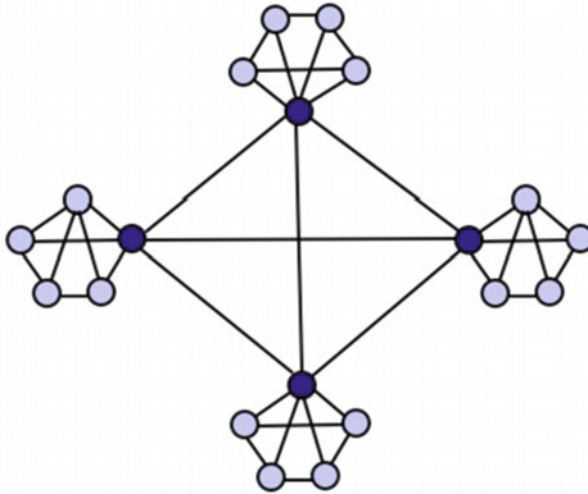


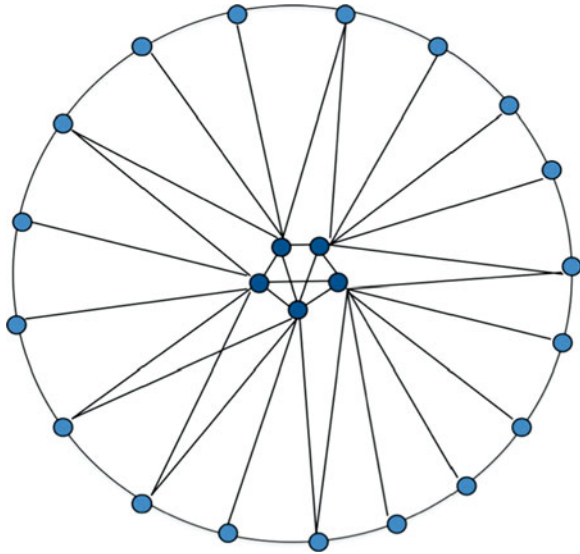
Fig. 28.4 Ideal-type for a pre-raiding guild

clearly keeping a guild alive and thriving is no small feat. Because of this, we decided to see if it was possible to predict guild survival using the variables in our data set. We started by taking two 1-month long samples from our data and marked guilds we had seen in the first sample but not in the second as dead, the others as survived. We then ran a logistic regression using survival as the dependent variable. Interestingly, we found that a lot of our social network metrics predicted a guild's survival with great accuracy.

Two different structures emerged for successful guilds depending on what stage of the game progression they were at. The tables with the corresponding metrics are available in Ducheneaut et al. 2007: below, we show “ideal-types” (in the Weberian (1949) sense) for each guild type. In the first guild type (Fig. 28.4), most interactions happen in small, densely interconnected subgroups loosely connected to each other. This is for guilds that have not yet reached the “endgame” with its large and complex raids. This fits the design of WoW quite well: these guilds are basically made of small bands of players leveling up through the game at the same pace and doing an occasional 5-man dungeon while maintaining a loose connection with each other.

For endgame guilds the picture is quite different – the network looks much more like a command-and-control structure, with a dense core of the most active players, most often the officers, all tightly connected to almost everyone in the guild to reflect the large amount of time they spend raiding together (Fig. 28.5). This is obviously quite a different picture, which brings us to a key issue in WoW's design: transitioning from one type of guild structure to the other is very hard. In a sense, this is the flip side of WoW's “two games in one” design – players spend most of their time playing casually with others and when they reach the endgame,

Fig. 28.5 Ideal-type for a raiding guild



they suddenly have to transform their group into something that looks more like the military. Unsurprisingly that is not for everyone (Williams et al. 2006) and a lot of guilds disband at that point.

Clearly, our data shows that raiding guilds are difficult to sustain at the endgame. There are probably multiple reasons but we would like to focus on one that might be reasonably straightforward to address: the almost complete absence of group support tools from the game itself. Picture the plight of a guild leader about to organize a raid: they will have to coordinate the activities of 40 players from perhaps 200 members in their guild, planning their actions in complex dungeons lasting several hours. And yet, the only tools at their disposal in-game are a guild roster and a calendar (and the latter is only a recent addition). While the lack of planning and coordination tool is only one facet of the guild survival problem, it is not hard to understand why guild leaders can burn out if that is all they have to help them. There is clearly a gap and in response, a small cottage industry of “guild portals” sprung up to support guilds in WoW. They can be quite useful, but they are Web-based, and therefore suffer from being disconnected from the game – not all data is available, and it is impossible to coordinate “in-world”, where it matters most. We mentioned earlier that Blizzard also allows players to develop user interface addons: while they can help coordinate groups a bit better from within the game, they are often bewilderingly complex and poorly integrated with each other, leading to another problem: information overload, which the screenshot below (Fig. 28.6) from a WoW raider’s screen illustrates.

Of course, we are not saying addons are the problem – in fact, they are often essential to high-end raiding. Still, there is probably a way to improve the situation. In particular, we think there is a real opportunity here for MMOG designers to think about more integrated “game groupware.” Anything helps, and they can start with



Fig. 28.6 A WoW raider’s user interface, overloaded with multiple addons

simple functions first like the calendar Blizzard added in 2008 to WoW. But eventually it should be possible to move to more complex tools: perhaps something like a dungeon planning tool with maps, ways of assigning guild members roles and positions, etc. Designers could also provide in-game “war rooms” where guildmates could meet “in avatar” and rehearse key parts of a fight. Additionally, they could also provide in-game guild stats and visualizations for guild leaders to monitor group performance, and the list goes on. However, our point is simple: based on our data, anything that makes managing a group easier would probably help guilds, which in turn would probably translate into a more satisfying experience for guildmates and guild leaders instead of the current high rate of burnout at the endgame. It is worth noting that there is a rich history of research in computer-supported collaborative work (CSCW) that could be leveraged here – it simply has not crossed over into the gaming world yet.

Another option could be to support guilds from the outside by monitoring their activities and helping them out when needed. We developed a small prototype called the Social Dashboard to illustrate what could be done in that domain (see Figs. 28.7, 28.8 and 28.9). In a nutshell, the Dashboard makes key variables in the guild survival model we presented earlier visible – for instance, one can see at a glance the average size of guilds on a given server. The widgets clearly mark thresholds in the model indicating when guild survival is threatened. A community manager at a company like Blizzard could start from this screen and then drill down to look at data for the most endangered groups. They could observe the evolution of key players in a guild’s social network over time, for instance, and perhaps offer incentives and rewards to

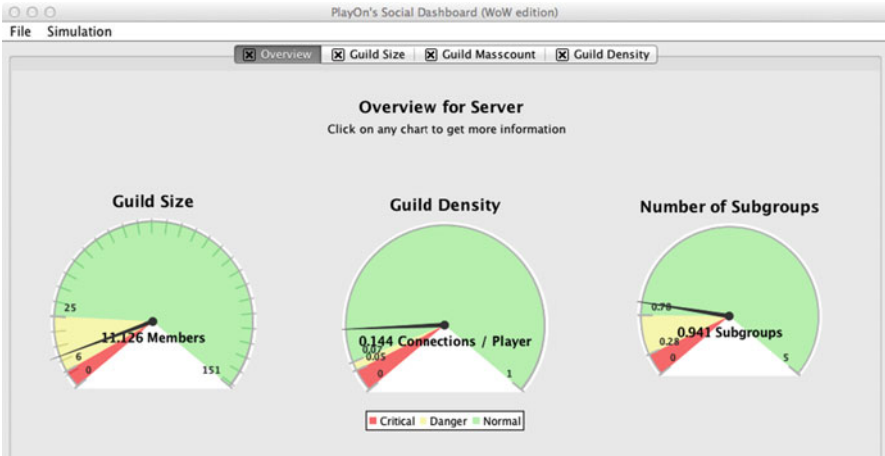


Fig. 28.7 The social dashboard. Three key properties of guilds (size, number of subgroups, and density) are under observation

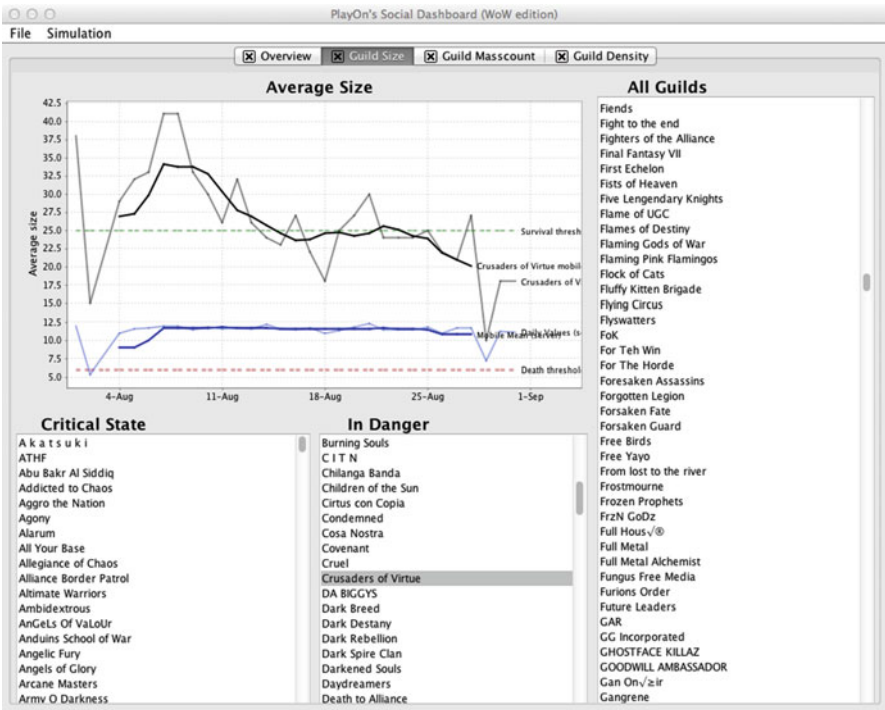


Fig. 28.8 The social dashboard. A variable (here, group size) has been selected for deeper analysis

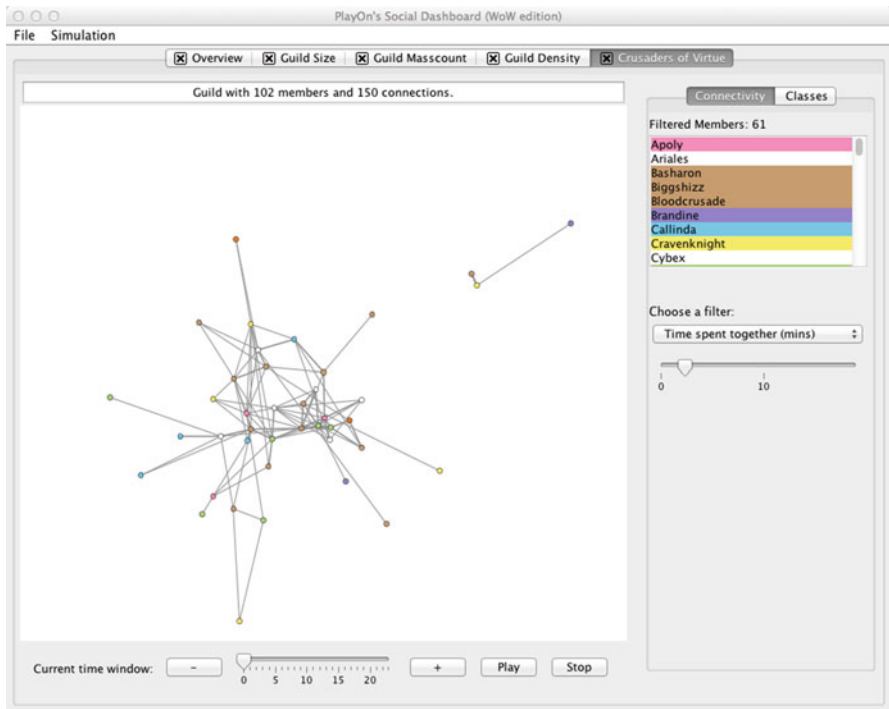


Fig. 28.9 The social dashboard. A guild’s social network and its evolution over time are being analyzed

the players most likely to keep the group together. Monitoring the “social health” of guilds in this way could perhaps help groups survive longer. More generally, it is another illustration of how player behavioral data could be used in profitable ways.

28.2.3 *The Expression of Personality in MMOGs*

Up to this point, our data and analyses were focused on illustrating various facets of social life in WoW. However, individual activities are equally important. As we were mentioning earlier, WoW has been very good at offering a very broad range of activities to please gamers with different preferences: for instance, players interested in competing against other players can do it through PvP combat, while players interested in more peaceful activities can work on individual quests to collect companion pets or earn titles based on how much of the world they have explored. This opens up the possibility of analyzing the relationship between a player’s personality outside the game and the kind of activities they prefer doing online. For instance, it could be that PvP combat maps back to aggressiveness offline – or not.

Table 28.1 Overview of Big 5 factors in personality

Factor	Adjectives: low scores	Adjectives: high scores
Extraversion	Quiet, reserved, solitary	Outgoing, energetic, gregarious
Agreeableness	Cold, critical, judgmental	Warm, friendly, compassionate
Conscientiousness	Careless, disorganized, spontaneous	Efficient, organized, neat
Openness to Experience	Cautious, conservative, practical	Inventive, imaginative, curious
Emotional Stability	Nervous, stressed, moody	Secure, calm, relaxed

Or it could be that intense exploration online maps back to openness and desire for new experiences. This is potentially interesting since it could help game designers tailor game experiences based on a gamer's personality profile.

In personality psychology, it is well-accepted that variation in personality can be captured by five broad factors (McCrae and Costa 1987). These five factors were developed using factor analytic techniques on adjectives and descriptive phrases of people culled from an English corpus. An overview of these five factors is presented in Table 28.1.

Studies in the physical world have repeatedly shown that judgments of personality of strangers are moderately accurate. For example, in face-to-face encounters, Extraverted individuals spoke louder, with more enthusiasm, and were more expressive with gestures (Funder and Sneed 1993; Kenny et al. 1992). Other research has shown that personality can be inferred from looking at someone's bedroom or office (Gosling et al. 2002). For example, Conscientious individuals had well-lit, neat, and well-organized bedrooms. And individuals who scored high on Openness to Experience had more varied books and magazines. This line of research has also extended to computer-mediated communication. In particular, studies have shown that moderately accurate personality impressions can be formed based on an individual's personal website (Vazire and Gosling 2004) or blog (Yarkoni 2010). For example, in terms of linguistic output on blogs, Agreeable individuals were more likely to use the first person singular, words related to family, and words related to positive emotions (e.g., happy, joy). These studies show that we leave behind personality traces in both the physical and digital spaces that we inhabit. Given that the average online gamer spends over 20 h a week in a virtual world (Yee 2006b), it is not difficult to imagine that some number of personality traces could be found in behaviors in an online game.

On the other hand, there are also reasons to believe that personality may not be readily expressed in virtual worlds. First of all, previous studies have largely focused on personality expression in everyday settings or linguistic output online. It is unclear how or whether personality is expressed via non-human bodies doing non-human things in a fantasy world (e.g., gnomish priests resurrecting the dead with magical light rays). And second, Turkle's argument of fluid identity experimentation (Turkle 1995) suggests that people could express or reinvent themselves idiosyncratically in WoW. If this were true, then we may not be able to identify consistent correlations between in-game behaviors and real world personality.

Thus, in a recent study, we sought to directly examine how well and in what ways personality may be expressed in a virtual world like WoW (Yee et al. 2011). Using our most recent data set, we examined the correlations between the self-report Big 5 personality factors and a set of player-level in-game metrics. A Monte Carlo simulation showed that our data contained almost eight times the number of significant correlations than would be expected from chance alone. Thus, there was a broad correspondence between the personality factors and in-game behaviors. Below, we summarize the significant correlation patterns identified for each personality factor.

- **Extraversion.** Aligned with the trait definition, players who score high on Extraversion preferred group activities. They participated in more dungeons, which require collaboration with other players. They have also completed a higher number of end-game 25-man raid dungeons. On the other hand, players who score low on Extraversion preferred solo activities, such as questing, cooking, and fishing.
- **Agreeableness.** Players who score high on Agreeableness give out more positive emotes, whether hugs, cheers, or waves. They also have a stronger preference for non-combat activities, such as exploration, crafting, world events, cooking, and fishing. On the other hand, players who score low on Agreeableness prefer the more competitive and antagonistic aspects of the game. They have killed more players, have more accumulated deaths, and focus more on getting better equipment. They have also participated in more player-vs.-player activities, such as battlegrounds, arenas, and duels.
- **Conscientiousness.** Players who score high on Conscientiousness seem to enjoy disciplined collections in non-combat settings. This is reflected in a large number of vanity pets, which must be collected over time. This is also reflected in high cooking and fishing scores, which reflect self-discipline in two activities that require a great deal of patience. On the other hand, players who score low on Conscientiousness seem to be more careless and are more likely to die from falling from high places.
- **Openness to Experience.** Players with a high Openness to Experience have more characters, and play on more servers. They also spend a larger portion of their time exploring the world and are more interested in non-combat activities, such as crafting and world events. Players low on Openness to Experience were more likely to focus on the combat-oriented aspects of the game, spending more time in dungeons and raids.
- **Emotional Stability.** While we found some significant correlations between this personality factor and some in-game metrics, these correlations were difficult to interpret as a whole. It is worth noting that previous studies have also had difficulty identifying meaningful behavioral correlates for Emotional Stability (Gosling et al. 2002; Mairesse and Walker 2006).

It is easy to imagine that VWs allow us to become whatever we want to be, but our findings show that even when we become orcs and elves, our personalities are still revealed. It does bear emphasizing how unique VWs are as a potential personality assessment tool. Most standardized personality assessment tools are highly reactive.

Asking someone whether they “like to insult other people” leads to a response that is highly influenced by response biases, and not the least of which is social desirability bias. Also, self-reports assessments capture these biased responses in only one moment of time, whereas VWs have the potential to generate longitudinal behavior profiles, aggregating hundreds of thousands of actual behavioral decisions over a 6-month period or longer. As other researchers have noted, we should study actual behaviors as they occur, in their natural settings as people go about their “humdrum lives” (Mehl and Pennebaker 2003). Given that the average online gamer spends over 20 h a week in an online game (Yee 2006b), we believe that VWs are a rich platform with which to explore these links between personality and behavior.

Our findings on personality inference suggest several possible applications. Personalized interfaces and system customization have long been of interest to the human-computer interaction community (Mackay 1991; Riecken 2000). In a system that is able to infer a user’s personality over time, several dynamic customization opportunities are possible. In a game environment, the system can dynamically prioritize presentation of personality-aligned game features. For example, an Extraverted player may be more engaged with a game that emphasizes social activities and social rankings. Or once a game infers that a player is low on Agreeableness, the system can make it more clear how to get started and engage in structured player-vs.-player content.

Another possible application directly applicable to online games, as well as other social applications, is using the inferred personality information to assist in the formation of groups. WoW currently uses a group formation system that largely selects players based on role, but such systems could be augmented to also take into account personality data. For instance, groups requiring a diversity of opinions might benefit from the inclusion of a wide range of personality types (Harper et al. 2007). In others contexts, such as in a structured player-vs.-player setting, a more homogenous mix might be more beneficial. And it is worth pointing out that we are not suggesting an automated system that would kick some players out of groups because they are low on Agreeableness. After all, the competitive nature of these players can be an asset in PvP settings, and an assertive nature can also be valuable to raid leaders. These concepts could also be applied in non-game contexts. For example, a social interaction system may dynamically assemble a more heterogeneous group for an initial brainstorming session to increase the diversity of opinions and ideas.

28.3 Conclusion

In summary, MMOGs like WoW are a kind of large-scale, persistent social engineering experiment that can be used to explore a wide range of issues, from sociability to group dynamics and personality. Their scale and the relative ease with which data can be collected make them ideal to study online social interactions quantitatively and the PlayOn project at PARC was launched to exploit this opportunity. While the scientific potential of games should not be underestimated, our

work also shows that social science research can suggest concrete ways in which future games could be improved. In this chapter, a limited overview of our findings led us to outline three possibilities – but there could, of course, be many more. The ones we talked about were:

1. Based on data revealing that social interactions can take many forms and might be less frequent (in the early game) than expected: encourage and support indirect social interaction – the “spectator experience”;
2. Based on data from social network analysis in guilds showing that transitioning from casual grouping to “hardcore” raiding is difficult: design “game groupware” to help group survive longer;
3. Finally, based on data about individual behaviors showing marked preferences for some types of in-game activities depending on the player’s profile: use behavioral traces to infer a player’s personality and dynamically adjust their game experience.

It is our hope that as MMOGs keep expanding and reach an ever-wider audience, game designers will be able to leverage data collection and analysis techniques like the ones we illustrated in this chapter to continue improving their product, leading to an even more satisfying online gaming experience for all players – including the authors.

About the Authors

Nic Ducheneaut: is a Senior Scientist in the Computer Science Laboratory at PARC. He uses a combination of methods (including data mining, surveys, and ethnographic observations) to study the social life of online communities. Based on these studies, he also designs and implements new computer systems to better support electronic communication and collaboration.

Nic’s most recent research focuses on the social dynamics of massively multi-player online games and virtual worlds: he founded the PlayOn project, which is conducting the longest and largest statistical study of player behavior in World of Warcraft to date (300,000+ characters observed over 5 years). Before that, he developed a wide range of novel social software ranging from email clients to recommender systems. He currently has 23 US patents pending and published more than 50 research papers in Human-Computer Interaction, Sociology, Communication, and Game Studies. Nic obtained his Ph.D. in 2003 from the University of California, Berkeley.

Nick Yee: is a research scientist at the Palo Alto Research Center (PARC). His research focuses on social interaction and self-representation in virtual worlds and online games. He is well-known for the Daedalus Project, a long-running online survey study of over 50,000 online gamers that examined who plays online games and why. As a graduate student at Stanford University, he conducted psychological

experiments to understand how virtual worlds allow us to break the rules of physical reality in productive ways. And at PARC, he has analyzed large-scale data sets of logged behaviors from online games. Nick is the author of more than 40 peer-reviewed publications in virtual environments and online games.

References

- Castronova, E. (2005). *Synthetic worlds: The business and culture of online games*. Chicago: The University of Chicago Press.
- Castronova, E. (2006). On the research value of large games. *Games and Culture*, 1, 163–186.
- Chen, M. (2009). Communication, coordination, and camaraderie in World of Warcraft. *Games and Culture*, 4, 47–73.
- Ducheneaut, N., & Moore, R. (2004). The social side of gaming: A study of interaction patterns in a massively multiplayer online game. In *Proceedings of CSCWW 2004* (Vol. 1, pp. 360–369), Chicago.
- Ducheneaut, N., & Moore, R. (2005). More than just “XP”: Learning social skills in massively multiplayer online games. *Interactive Technology and Smart Education*, 2, 89–100.
- Ducheneaut, N., Yee, N., Nickell, E., & Moore, R. (2006). Alone together? Exploring the social dynamics of massively multiplayer games. In *Proceedings CHI 2006* (pp. 407–416), Canada.
- Ducheneaut, N., Yee, N., Nickell, E., & Moore, R. (2007). The life and death of online gaming communities: A look at guilds in World of Warcraft. In *CHI 2007 proceedings* (pp. 839–848), San Jose, CA.
- Dunbar, R. (1993). Coevolution of neocortical size, group size and language in humans. *Behavioral and Brain Science*, 16, 681–735.
- Funder, D., & Sneed, C. (1993). Behavioral manifestations of personality: An ecological approach to judgmental accuracy. *Journal of Personality and Social Psychology*, 64, 479–490.
- Gosling, S., Ko, S., Mannarelli, T., & Morris, M. (2002). A room with a cue: Judgments of personality based on offices and bedrooms. *Journal of Personality and Social Psychology*, 82, 379–398.
- Harper, F., Frankowski, D., Drenner, S., Ren, Y. Q., Kiesler, S., Terveen, L., Kraut, R., et al. (2007). Talk amongst yourselves: Inviting users to participate in online conversations. In *Proceedings of IUI 2007* (pp. 62–71). Honolulu, HI.
- Kenny, D., Horner, C., Kashy, D., & Chu, L. (1992). Consensus at zero acquaintance: Replication, behavioral cues, and stability. *Journal of Personality and Social Psychology*, 62, 88–97.
- Lazarro, N. (2004). *Why we play games: Four keys to more emotion without story*. In GDC. Presentation at GDC 2004, San Francisco, CA.
- Mackay, W. (1991). Triggers and barriers to customizing software. In *Proceedings of SIGCHI 1991* (Vol. 1, pp. 153–160). New Orleans, LA.
- Mairesse, F., & Walker, M. (2006). Automatic recognition of personality in conversation. In *Proceedings of the human language technology conference* (Vol. 1, pp. 85–88), New York, NY.
- McCrae, R., & Costa, P. (1987). Validation of the five-factor model of personality across instruments and observers. *Journal of Personality and Social Psychology*, 52, 81–90.
- Mehl, M., & Pennebaker, J. (2003). The sounds of social life: A psychometric analysis of students’ daily social environment and natural conversations. *Journal of Personality and Social Psychology*, 84, 857–870.
- Moore, R., Ducheneaut, N., & Nickell, E. (2007). Doing virtually nothing: Awareness and accountability in massively multiplayer online worlds. *Computer Supported Cooperative Work*, 16, 265–305.
- Reeves, S., Benford, S., O’Malley, C., & Fraser, M. (2005). Designing the spectator experience. In *Proceedings of CHI 2005*, (pp. 741–750), Oregon, WA.

- Riecken, D. (2000). Personalized views of personalization. *Communications of the ACM*, 43(8), 26–28.
- Taylor, T. L. (2006). *Play between worlds: Exploring online game culture*. Cambridge: The MIT Press.
- Turkle, S. (1995). *Life on the screen: Identity in the age of the internet*. New York: Simon & Schuster.
- Vazire, S., & Gosling, S. (2004). e-Perceptions: Personality impressions based on personal websites. *Journal of Personality and Social Psychology*, 87, 123–132.
- Weber, M. (1949). *The methodology of the social sciences*. New York: The Free Press.
- Williams, D., Ducheneaut, N., Li, X., Zhang, Y., Yee, N., & Nickell, E. (2006). From tree house to barracks: The social life of guilds in World of Warcraft. *Games and Culture*, 1, 338–361.
- Yarkoni, T. (2010). Personality in 100,000 words: A large scale analysis of personality and word use among bloggers. *Journal of Research in Personality*, 44, 363–373.
- Yee, N. (2006a). The labor of fun: How video games blur the boundaries of work and play. *Games and Culture*, 1, 68–71.
- Yee, N. (2006b). The demographics, motivations, and derived experiences of users of massively multi-user online graphical environments. *Presence: Teleoperators and Virtual Environments*, 15, 309–329.
- Yee, N., Ducheneaut, N., Nelson, L., & Likarish, P. (2011). Introverted elves and conscientious gnomes. The expression of personality in World of Warcraft. In *Proceedings of CHI 2011*, 753–762, (Vol. 1, pp. 753–762), Vancouver, Canada.

Chapter 29

Designer, Analyst, Tinker: How Game Analytics Will Contribute to Science

Edward Castronova, Travis L. Ross, and Isaac Knowles

Take Away Points:

1. Game designers use massive player data sets to analyze player behavior
2. Their questions are relevant for science
3. Their methods are very good, often better than those available to researchers

29.1 Introduction

If you were watching the world of science in 1835, you would be fairly certain of one thing: we would never know what stars are made of. By that time scientists had discovered that the stars were quite far away and they could not imagine any technology that would take humans from this planet to those places and back again.¹ Then one day, Robert Bunsen (of the burner) and his colleague Gustav Kirchhoff noticed that when elements were placed in a hot fire, the light they emitted proved to have a signature distribution of color on the spectrum. A spectrograph observation of burning helium looks different, uniquely and reliably, than one of burning nitrogen. In the instant of discovery, the two scientists had learned how to sample the stars.

It was an accidental discovery.² It happened not because the scientists developed a deep theory and then tested it, but because they were fiddling around (in an

¹ Not only that, but in the effort to sample a star's materials, a scientist would almost certainly suffer burned fingers.

² <http://www.aip.org/history/cosmology/tools/tools-spectroscopy.htm>, observed January 31, 2012.

E. Castronova, Ph.D. (✉) • T.L. Ross, Ph.D. • I. Knowles, Ph.D.
Department of Telecommunications, Indiana University, 107 South Indiana Avenue,
Bloomington, IN 47405, USA
e-mail: castro@indiana.edu

intelligent way) with a research tool.³ This process of fiddling around with research tools is something that game developers are doing with increasing regularity. Developers often experiment with different game designs by making changes to game worlds and observing the resulting changes. Tinkering of this nature is not limited to large-scale game worlds like MMOs. Designers tinker in multiplayer games of various sizes; controlled tinkering with telemetry has the opportunity to lead to interesting insights across a range of games. To a social scientist focused on the careful development and testing of theory, it may seem surprising that this type of intelligent tinkering is poised to lead to dramatic advances for science. To a game developer this type of tinkering may not seem surprising at all and no different from standard game design.

Yet, we contend that this chapter presents topics of importance for both the social scientist and the game developer. The first section is for social scientists. Like the argument made in Chap. 28, we argue that multiplayer games will be the primary location for scientific discovery in the coming years and that social scientists may want to look toward the game industry for new theoretical and methodological insights. The second section is for game designers; it presents a method for measuring a dependent variable of interest among different populations of players – engagement/fun. The third and final section explores a case where tinkering can be simultaneously valuable to social scientists and game developers. It examines how existing social scientific theories of social norms and institutional design can help designers steer player behavior, and how controlled tinkering lets game designers add to and test social scientific theory.

29.2 Tinkering for the Social Scientist

Imagine the ideal research tool for tinkering with societies. For most of social science, the relevant questions are about the beehive, not the bee. Sociologists are concerned with national birth rates, not individual motivations to have children. Economists wonder what determines the overall trend in the stock market, not how to make a killing from information about a company's products. And so on. Thus, the machines social scientists would want are beehive manipulators – tools for testing, probing, and measuring human society. Put another way, while psychologists and neurologists can measure brain scans and observe a few individuals as they perform a laboratory task, scientists of society need to scan large groups of brains and observe thousands or millions of individuals as they go about their behavior. Tinkers need to be able to feel free to explore, therefore the ideal tool should also be cost effective. The tool itself might be expensive to build, but once built, it should allow for relatively inexpensive manipulations. If the cost of a manipulation is too expensive careful planning takes over and exploration for the sake of pushing the boundaries is lost.

³ We acknowledge that there are also theoretical tools in science, but we are limiting our discussion of tools to research tools.

Of all the existing tools available for studying societies, none of them possess all of these features. In fact, as the limits of social scientific tools have been explored, it appears that certain tradeoffs must be made. Some of these tradeoffs lend themselves more readily to tinkering than others.

Historical analysis, direct observation and data-mining can examine society at the social scale; in the case of natural experiments, they can even provide semi-controlled trials. But they can't be changed or altered. They are not tools that allow tinkers to explore what-if questions.

Laboratory and field experiments can be manipulated, they provide controlled manipulations, and they work on real humans; however, they are difficult to scale. Some very impressive work is increasing the scale, but the fact remains that these experiments are still limited in scale and the cost for a single manipulation is prohibitive of tinkering (Beaman et al. 2012).

Computer simulations are ideal for tinkering. They can examine large scale interaction, and they are easy to manipulate and explore. The problem is that they can only explore theoretical models. Computer simulations do not use real people and so they don't allow tinkering and observing actual human society. With computer models, the tinkering is still theoretical.

Of course, it is not reasonable to expect a tool to be perfect. There are no perfect tools. All tools have limitations. This is why good science uses multiple tools to test a theory. Tools can complement one another. However, given the state of the art of current tools in the social sciences, tinkering is mostly a theoretical practice. Tinkers currently explore mathematical models and computer simulations in order lend insights about the origin and reason for human behavior, and many interesting insights have come from tinkers doing this very thing (Axelrod and Hamilton 1981; Rendell et al. 2010; Schelling 2006). However, without a tool that can allow relatively cost effective tinkering with real human beings, tinkering will not have the same impact it has had in the natural sciences.

29.2.1 The Role of Games in Tinkering

Enter games. Today, large-scale multi-player games can host millions of players. The conditions of interaction are completely fabricated by the designers. In some games, players do not interact with other people at all. In others, players collaborate and compete in an incredibly wide variety of arenas, forums, and institutions. Many of these interaction opportunities are modeled on, and in some cases perfectly replicate, ones found in the real world.

Because the entire environment is designed from the ground up, a game world is extremely malleable. Open for tinkering, you might say. And game designers do tinker. The game World of Warcraft, launched in 2004, has for years been subjected to major or minor patching, and it currently changes at the rate of about one patch per month. As of this writing (January, 2012), there have been more than 80 distinct versions of World of Warcraft. Design teams are constantly optimizing for user experience, and in doing so, they are constantly learning about the societies in their games.

This learning has become a formal part of game design practice, as the chapters in this volume will attest. It is now standard practice to make small changes and then test them on a random sample of the player base. By creating comparable conditions and subjecting one condition to an intervention, designers can determine how the intervention changes the game and its society. This protocol, also known as “A–B testing,” (see Chap. 4) is of course no different from the standard controlled experiment protocol of the natural sciences. It is a simple and powerful mode of discovery.

Another tool of discovery is machine learning (see Chap. 12). Here, the analyst searches for patterns in an existing dataset using a search algorithm. The search algorithm is given a set of parameters and determines the existence of relationships between those parameters. This use of machine learning is no different from its use in a research project in the natural sciences. However, there are several critical differences with ordinary scientific research.

In game analytics, the research populations are huge and the environment is under the control of the designer. Moreover, game designers are not trying to discover general knowledge; they are trying to make money. Yet, even with these differences, we argue that game analysts are likely to make discoveries of major importance to a general understanding of people and society.

While game designers are not necessarily pursuing knowledge for its own sake, the knowledge they are pursuing is of general importance. The game industry wants to make money, and usually this means, the industry is driven to make their customers happy. Games are built on a careful combination of intrinsic and extrinsic motivation. Thus, the tinkering that game analysts will do will be in the service of discovering knowledge about human motivation. This occurs on two levels. On the micro level, designers want to know how individual people react to game design decisions. On the macro level, they want to know how social systems react to those decisions.

Are these questions facing game designers the most important questions we face today? Let’s think about that. When a game designer queries a human population, what does he ask? Basically, he wants to know what makes the group happy. The game designer’s job is to provide a satisfying experience to the users. Because the requirement is plural – users, not user – this is a social goal. The game designer has to balance the happiness of any one player against the happiness of others. The designer’s objectives are expressed in terms of aggregates – total revenue, total time in-game, total churn. The designer has to think in terms of the group as a whole. At the same time, he has to be aware that these group effects emerge from the experiences and decisions of thousands or millions of individuals, each of whom has a completely personalized encounter with the game he has made.

Is this not exactly the same problem faced by a well-intended head of state? Assume a good king. The king wants to make his people happy. But this is difficult. They are free, and they rightfully pursue happiness in their own way. Yet, one man’s happiness bumps up against another. His people squabble with one another. They can be dangerous. Under these circumstances, by what rules should our king govern? And given any set of rules, by what means does our king know that the rules he has emplaced are the best?

Facing this set of informational problems, our good king would want to build many little kingdoms and maintain them as test beds for his policies. He would then want to observe things like the migration rate to and from his realm, the time spent actively there, and the willingness (if any) of the people to support his rule through their taxes, time contributions, and verbal expressions.

This knowledge of statecraft is precisely what massive multiplayer game design seeks. In fact, it is the central goal of game-makers to answer the question: How can one design an environment, in full respect of the autonomy of individuals, in such a way that their motivations, acted upon, lead them into satisfied, happy experiences? But while good governance and good game design share many of the same goals, they emphatically do not share the same tools. Even in research universities, where questions of public policy are studied with the most advanced research tools available, the common practice is to examine policy-relevant questions by doing experiments on 20–50 college undergraduates who have done some exercise in a lab for a few hours on a Thursday afternoon. Meanwhile, game designers test their questions with millions of players over the course of weeks, months, even years.

Because they are studying similar questions but using vastly superior tools, game designers are likely to break open the frontiers of knowledge first. Game designers will know far in advance of others how to encourage people to follow their hearts and yet obtain happy outcomes. Game designers will know more about making good markets than anyone else. They will know more about making good governments than anyone else. They may well make the “best” societies (however defined) that humanity has ever known.

29.2.2 Some Provisions for Academics

All tools have limitations. Games are not a panacea for the social sciences. Here are four particular features that could limit their impact on research.

The first is plain enough: the game-playing demographic is not representative of any existing country’s population. For example, we know from several surveys that the demographic of U.S. game players is relatively younger, whiter, more/better educated, and wealthier than the average U.S. resident, and that by far more men than women play video games (Entertainment Software Association 2011). Yet, we do not know what this means for extrapolation of gamer behavior to the rest of the world. Can broad claims be made? What findings are generalizable? While some might be willing to give this problem a pass, any student of human behavior would be skeptical of claims based on an unrepresentative population.

The second important difference between games and the real world is the freedom of the player to enter and exit. Economists have made some headway with field tests of theories, using populations from real-world developing countries (Beaman et al. 2012). In these experiments, the stakes that subjects face are real, because switching costs are prohibitively high. Players, however, are not so limited in their course of actions; if a player does not like the hand she is dealt, she may switch to

another game or another virtual world. This raises both attrition and self-selection problems in studies of human behavior that use gamer behavior.

The third problem is that the game environment is constructed based on our best knowledge about how the real world functions. Thus, a test of a policy, or of a particular institutional arrangement, will actually be a test of how players react to a simplified – and likely flawed – representations of those arrangements. In his own work, Dmitri Williams (2010) calls this the “mapping problem”, and it is indeed a conundrum that requires careful consideration during research design stages.

The fourth problem that virtual worlds face is that the societies within them are dynamic systems. They change over time and the previous state of the system impacts the current state. Therefore, it is difficult to determine if an outcome has occurred due to an earlier fluctuation in behavior. The entire idea of tinkering is based on random experimentation, but in complex and chaotic systems, we must ask whether the outcome observed occurs every time, or if what we observe is rather a property of the somewhat random interactions between individuals that preceded the change. How many virtual worlds must be run in parallel for us to be confident of our empirical claims? Is running parallel experiments feasible? This is an important practical concern.

Fifth, most virtual environments today offer a much narrower scope of action than does the real world. Of course, experimental environments in the real world also narrow the scope of action considerably. Research subjects are expected not to leap out the window. In virtual worlds, the window is simply impassable! Nonetheless, the nature of the scope of action is an element of any experiment and there are substantial restrictions on what one can and cannot do in a given virtual space.

Finally, today, the consequences of virtual action are smaller than the consequences of real-world action. That may not always be so, but for the moment, one cannot cut one’s finger through a choice in the virtual world.

To sum up, games are created by limited beings using a limited technology, played by a limited set of individuals with an unlimited set of other uses for their time. This is games’ greatest weakness as research tools. However, it is also their greatest strength. Because we can identify the weaknesses, because we know the rules, because the rules are simple and easy for players to understand, we find it possible to argue that the behaviors of players – micro and macro – are very accurate reflections of how they would behave in real life if faced with those same rules. Therefore, despite the chasm that separates our games from our lives, we can extrapolate from gamer behavior to real-world human behavior with a fair degree of accuracy.

While the current potential of games as research tools has by no means been tapped, advances in social science research using games will depend upon advances in game design. Conversely, advances in game design will come, in part, with advances in micro- and macro-social research. As both game analysts and game researchers become better practitioners, and as our understanding of how our societies function improves, so can games not only become more fun, but also become better reflections of our world.

29.3 Industry-led Tinkering: Is It Worth the Risk?

For researchers studying political, cultural, and economic dynamics, being able to watch an entire society evolve in response to changes in control variables is justifiable despite the limitations. For game designers, however, causing these changes is fraught with risk precisely, because we know so little about how societies – including game populations – will respond to behaviors in light of minor modifications to the world around them.

To consider a concrete example, suppose that one multiplayer map in *Call of Duty – Modern Warfare 3* (Activision 2011) were modified in a minor way (perhaps by removing some cover) to weaken a common position taken up and defended by players that enjoy sniping others from a distance. Suppose that this incentivizes a switch from long-range rifles to assault rifles in that level. The way players and teams approach that particular level will then change considerably: new positions become tenable, others become obsolete. New strategies and tactics must be learned, adapted, and deployed. Meanwhile, increased usage of assault rifles in the game, improves player skills and abilities with those guns, and speeds up access to new accessories for assault rifles, while slowing skill and accessory acquisition for sniper rifles. This could change the dynamics in other game levels. It will make certain types of achievements easier to obtain and others more difficult. All of these changes, which could result from a simple environmental change in a single level of a first-person shooter, will affect player satisfaction. It could mean the difference between buying a game and renting it. It also has implications for the supply of used games, which will have multiplicative effects on revenues. One simple change could mean a difference for the bottom line in the amount of hundreds of thousands of dollars.

Sounds risky. Nevertheless, consider this: we have been using the word “tinker” to evoke the idea of creative exploration of the way that players interface with the game and, in the case of multiplayer games, how players use them to interface with one another. However, for game developers, it is worthwhile to think about tinkering as research and design using existing games and player bases. In a sense, we are suggesting that for a game designer, the current publication’s release version should be the next publication’s pre-alpha version.

R&D (Research and Development) of this sort is risky and expensive, and seems to be rare in the game industry. This is not to say it is never done – for what is game analytics, if not the study of player activities with an eye to obtaining or sustaining revenues? Rather, this R&D involves few players, or it involves all players in a game, but is completely passive. In other words, R&D of the type we are considering is distinct – from that described by, say, Ben Lewis-Evans (2012) – because it involves subjecting large swathes of the player base to controlled experimentation.

The benefits and costs of tinkering will depend on the nature of the change, the nature and complexity of the game being changed, and on the platform(s) on which the game is played. The ability to tinker is also obviously limited to game designers with access to current release versions and their players. Finally, any successful R&D of this type depends upon high-quality embedded data collection

methods within the game, itself. These limitations aside, tinkering should be seen as an investment – inherently risky, but likely to turn up some valuable gems of new and unique knowledge that helps developers produce better games.

In what follows, we address two very different matters that both relate to tinkering. The first of the two sections asks: “if we change the design of a released game – if we tinker with the rules, the environment, or what have you – how do we measure the effect of the change on player enjoyment? And how can we tell how well players are adapting to the introduced change?” For developers, these questions are of obvious importance. Good decisions require good analysis, and we think that the so-called stochastic frontier models of ideal player behavior that we introduce are a useful addition to the game analyst’s toolbox. Games researchers will also see the value of this modeling technique for its applications in measuring learning in a game environment, and for its use in making comparisons with real-life organizational behavior.

After that, we consider a proposition for tinkering that should intrigue designers and researchers, alike. In particular, we look at the role of social norms in determining player enjoyment, and discuss the possibility of introducing or shifting norms in gamer populations. The value to social science researchers is self-evident. The value of this discussion to game designers is also quite high, in that: players of both single and multiplayer games derive at least some of their enjoyment from participation in the community.

29.4 Stochastic Frontier Analysis for Measuring Player Effectiveness and Fun

In this section, we propose to measure engagement/fun based on a particular conception of the player rooted in economic theory. We view the gamer as an entity whose “job” is to take their free time and their tools to produce a single output: fun. A game-designer provides tools that individuals purchase and then utilize to generate as much enjoyment as they can in the time they have available. It is useful to conceive of a player in this way for several reasons. First, it enables mathematical modeling of player behavior. Moreover, as we hope to convince the reader, it enables comparisons of individual players that can help clarify how effectively they make use of the games they play to create personal satisfaction. If a metric of player effectiveness can be produced, then it can shed light on several questions relevant to both academia and the industry.

Briefly, let us consider more carefully why it could be useful to conceive of a player as a worker, and what it means to have fun. On the latter point, we have very little to say. Some guidance is available from MMO researchers who make heavy use of Bartle’s (1996) player motivations model and its extensions, e.g. Yee (2006). Still, what is fun is player-dependent and context-specific, though it likely correlates strongly with several different readily measurable outcomes for players, depending on the game. On the former point (player-as-worker), it seems obvious that games

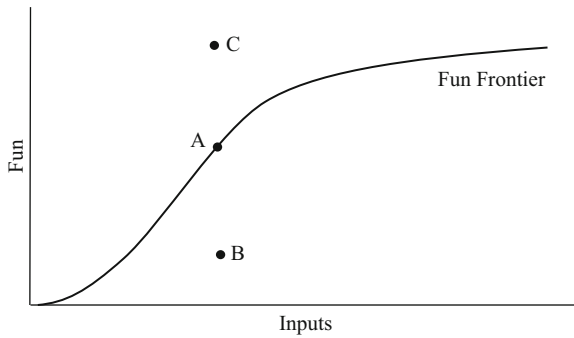


Fig. 29.1 A Fun Production Frontier. As a designer adds more intensity or length to an experience (the inputs), the maximum possible fun rises rapidly at first and then tapers off. The *line* shows the maximum possible fun that can be produced for a given amount of inputs. Inputs are costly, and therefore the designer would hope to achieve this maximum of fun. From this perspective, a point like *B* is inefficient – for the same inputs, and input cost, the designer might have achieved the level of fun indicated by *A*, which is greater (Presumably this leads to greater revenues as well.) Points like *C* indicate what is not possible. The inputs that produce the level of fun indicated by *A* simply cannot produce the level indicated by *C*. The level *C* is unattainable, but *A* certainly is attainable, and the designer ought to avoid *B* since more fun can be created for the same effort. The Fun Production Frontier helps distinguish between achievements that are achievable with great effort (moving from *A* to *C*) and achievements that require no more effort but just a better reorganization of effort (moving from *B* to *A*)

must be played, and playing takes time and resources away from the pursuit of other activities. Thus playing, like any other activity, can be construed as a productive process leading either to some intermediate input to another activity, or to a final outcome. To that end, a player is working for herself.

We do not wish for this chapter to become a course in economics. However, to understand the foregoing, some introduction to production theory is in order. As a first approximation, economists assume that firms maximize profit by producing on the so-called “frontier of production” – where the utilization of all time, money, materiel, and effort are used in proportions that maximize profits (Kumbhakar and Lovell 2000). By analogy, we can apply this paradigm to our idealized player. Figure 29.1 is a simple diagram that shows a hypothetical production frontier for player fun. To simplify exposition, suppose that for any particular game, there is just one input required to produce fun. This is on the horizontal axis. The amount of fun produced is represented by the vertical axis. Point *A* represents the ideal situation for both player – and for the game designer – where the player produces as much fun as she can with the inputs she has available. In real life, of course, it is unlikely that the player is always using best practices in the fun production process, and so the actual amount of fun created by the player could be less than its potential for a given amount of input. This is denoted by point *B* in Fig. 29.1. Points below the frontier are achievable but suboptimal. Points beyond the frontier, like point *C*, are not achievable.

These properties of the production frontier lend themselves to mathematical expression in the form of a production function. Consider the following equation:

$$Y_i = f(K, L) = A_i K^\alpha L^{1-\alpha}, \quad 0 \leq A \leq 1 \text{ and } 0 < \alpha < 1$$

This is known as the Cobb-Douglas (CD) production function. Y_i is the output of production (fun) for player i , K and L are inputs (tools and time, respectively), and A_i and α are parameters to the model. A_i expresses the degree of efficiency with which K and L are utilized, where $A_i = 1$ is most efficient. The parameters α and $1-\alpha$ are the so-called “output elasticities” of K and L , respectively. This means that a 1% increase in K (or L) leads to an α (or $1-\alpha$) percent increase in Y .

A few more brief points are in order. First, note that there are a multiplicity of ways to express production; for example, we could let all of the productive factors enter additively, as in $Y_i = A_i \beta_1 K + A_i \beta_2 L$, or we could express fun production as $Y_i = A_i \times \min \{K, L\}$. There are innumerable conceivable production functions; the CD function is but one special case of one commonly used class of functions.⁴ We (and many economists) use the CD function because it conforms to some stylized facts about production. In the context of this paper, one property is that you must have at least some of every input to produce fun: you need to devote time to play, and of course, you need to have the game. Another property of the CD function is that it demonstrates constant returns to scale – a doubling of every input exactly doubles the amount of fun. A third property is that, by construction, the producer reduces one input by 1% and increase the other by 1% while keeping output constant. Finally, it demonstrates diminishing returns to either input, holding the other input constant. In other words, if you wanted to double your fun only by spending more time playing, you would have to more than double your play time.⁵ Thus, our chosen function represents a particular hypothesis about fun production.

We would like to take the data to this function to estimate the parameters. The workhorse in the economist’s tool box is simple linear regression (Greene 2012, Ch. 2). To estimate the parameters for our Cobb-Douglas production function, we first take logarithms of both sides and then add an error term to obtain:

$$\log Y_i = \log A + \alpha \log K_i + (1 - \alpha) \log L_i + \varepsilon_i$$

Assuming that errors are symmetric and independently and identically distributed, and given that both the inputs and output are strictly positive, this is a simple estimation problem. However, it ignores the “frontier” nature of the production of fun. The actual value of individual efficiency of effectiveness in this model will be

⁴See (Mishra 2007) for a lengthy discussion of different production functions in the context of the history of economic thought. Much more complex production functions are possible, but we eschew deeper discussion in the interest of clarity of presentation.

⁵This is easily shown by setting $2f(L_1, K) = f(L_2, K)$ and solving for the ratio of L_2 to L_1 , which will be strictly greater than 2. For a discussion of these properties, including an concise introduction to production theory in economics and the properties of the Cobb-Douglas function, in particular, see Mas-Collel et al. (1995, Ch. 5).

unidentifiable as such. The parameter α will also be estimated as if we were looking at the average player. This is all well and good, but we are interested in the best way that a player can have fun with the given inputs, and we want to know the relative ability of our players to have fun. We also want to know how the player, having the most fun with her given inputs, is combining those inputs.

In order to obtain this information, we can use an estimation technique known as stochastic frontier analysis (SFA) to estimate the value of A_i – the efficiency parameter – for each player. Such a metric would be valuable in that it could tell a developer how well the player is using the game to produce fun relative to his fellow players. Or it could tell an academic how well a player has learned to play a game, which could be useful in a number of research settings.

Aigner et al. (1977) (hereafter, ALS) introduced SFA. When the frontier is of interest, they proposed that instead of one error term, we use two: one for idiosyncratic variation with mean zero, and another restricted to nonnegative values to represent variation in the efficiency of each observed unit. This leads to the following specification:

$$\log Y_i = \gamma + \alpha \log K_i + (1 - \alpha) \log L_i + u_i - v_i$$

where now $u_i \sim N(0, \sigma^2)$; the efficiency term, v_i , is drawn from some distribution with a strictly positive domain (for example, the half-normal distribution); and γ is a constant term. One would estimate this model using the technique known as “maximum likelihood estimation,” which, happily enough, also provides estimates of the variance of the efficiency term. Once the estimation is complete, the analyst can obtain measures of relative efficiency for every individual in the sample.

In the interests of space and clarity, we will not comment further on the SFA methodology. Readers who are interested in the statistical techniques will find useful introductions in any advanced econometrics textbook; see Greene (2012, pp. 839–845). For a lengthier motivation and exposition of SFA, see Kumbhakar and Lovell (2000). William Greene’s course website (<http://people.stern.nyu.edu/wgreene/>) provides exhaustive literature reviews, discussions, and extensions.

We now consider what might constitute measures of “tools” and “fun”. First, there need not be just one “tool”. There can be a set of tools, each vital to the production of fun used in different relative quantities. Usually, we are interested in the set of tools that differentiate players. Some of these tools are external. A game that is played on multiple platforms is played using different tools depending on whether it is a PC or console version, for example. Most tools are internal to the game, however, and we want to measure the quantities in which they are used. Some tools are simple choices – whether or not the player inverts the Y-axis on her controller, for example. Some tools are accumulated during the course of game play, such as the number of cities founded or buildings produced in an RTS, or the number or quality of weapons used in an action game. Another tool that is accumulated is player experience and skill. Still other tools could be based on social interaction: the size and density of one’s social network can be an extremely important tool for having fun in a game.

Which brings us to fun. How can we measure it? In a few cases, one of which we discuss in detail below, player motives are heavily focused on a single outcome. If the set of players being analyzed are chosen carefully, or if the game being analyzed is sufficiently simple, it is easy to argue that performance along that outcome is a good proxy for fun. But most of the time, any single metric will be a very noisy signal, and modeling is the multivariate problem that most analysts expect. In other words, what we are proposing here is not a way of simplifying life for the analyst; rather, it is a method to get at new information to which the analyst may not previously have had access.

29.4.1 An Example of Stochastic Frontier Analysis

To show the use of SFA in practice, we now consider a simple example. In particular, we apply SFA to the production of boss kills by leading US guilds in the game World of Warcraft (hereafter, WoW). This research was initially motivated by our interest in how well and how quickly guilds learned to overcome the various challenges in the raiding instance. We decided that SFA could be a fruitful approach. We later came to see that the efficiency term could be interpreted as a measure of how well the guilds produced fun for their members. Though controversial, our argument for this conceptualization is that highly competitive raiding guilds focus intently on obtaining world- or region-first boss kills. Casual observation suggests that (a) the goal of playing a game is fun, and, moreover, (b) in the context of guild raiding the goal of guilds – the thing that produces the fun – is world- or region-first kills. Given these assumptions, a good measure of fun is the number of boss kills. It is worth noting, in light of the above discussion of fun, that we would never, ever suggest using boss kills as a sole metric of fun in the broader population of MMO players, nor for the populations of any other games. We focused on this very small group of players because data were easy to obtain and because this focus simplified measurement; we do not wish to imply that boss kills is a preferred metric of fun for most players in most games. But it is an excellent metric for the fun had by highly competitive raiders. We also know, again from casual observation, that there are some (perhaps many) exceptions to the general assumption that raiding guilds raid for the prestige of getting boss kills. Our analysis focuses on guilds that quite clearly are in the game to kill bosses. Therefore, the results do not pertain to guilds whose main objective is to socialize, dance, role-play, challenge dominant capitalist media tropes, stare at virtual navels, or the many other activities that a guild may do. Many such guilds exist (Taylor 2012; Witkowsky 2012), they are however not the subject of this particular study.

29.4.1.1 Data Collection and Description

During the summer of 2011, Blizzard Entertainment released Patch 4.2 for WoW. Entitled “Rage of the Firelands”, it introduced a new raiding instance with seven bosses, the final boss being the fire creature called Ragnaros. Using our own WoW

account, as well as the Blizzard Community API,⁶ we automated the collection of information on the quality of weapons and armor carried by guild members and on the amount of time each member spent in the Firelands, the new raiding instance. We chose to focus our attention on the top 30 US guilds, where rank was determined by the date on which the guild completed the previous tier of raiding. The data on rank were gathered from the website WoWProgress (wowprogress.com). Since both 10- and 25-man versions of the raid are available, we chose to focus only on those guilds that participated in 25-man raids. During the week of June 12th to June 19th, we queried each guild's server about once every half hour during normal raiding hours (about 5 pm–5 am EST) as to the online status of each member of the guild. If online, we recorded each guild member's level and location. Twice a day, we used the Blizzard Community API to gather information on the item level of each piece of each guild member's weapons and armor.⁷ Data on bosses killed were also gathered from WoWProgress. At the time of the data collection, some guilds had defeated the sixth boss in the Firelands, but none had yet defeated Ragnaros.

To construct our measure of time spent raiding, we aggregated the total elapsed time spent in the raiding instance of the 30 guild members who spent the most time in that instance. To construct our measure of gear quality, we took the unweighted average of the equipped item levels on each of those same guild members.⁸ Our measure of progress was the number of bosses in the instance, out of eight, that the guild had beaten as of June 19th.

29.4.1.2 Our Results

We estimated the following model:

$$\log \text{Kills}_i = \gamma + \alpha \log \text{ItemLevel}_i + (1 - \alpha) \log \text{RaidTime}_i + \beta \text{PVP}_i + u_i + v_i$$

where PVP is an indicator variable for whether PvP combat is unrestricted on the particular server on which the i -th guild played the game. The results of our estimate are reproduced in Table 29.1. The estimated value of α is 0.342. This suggests that a 10% increase in the average item level of the guild's weapons and armor would lead to a 3.42% increase in bosses killed, while a 10% increase in total time the guild spent in the raid would lead to a 6.58% increase in bosses killed. We think that one of the most interesting results is the coefficient on PVP, which is consistently negative across many different specifications beyond the one reproduced here. Our results suggests that, holding other factors constant, guilds on PvP servers had completed 20% fewer bosses in the Firelands than guilds on PvE servers. We suspect that

⁶For more information, start at the appropriate forum: <http://us.battle.net/wow/en/forum/2626217/>

⁷The item level of a weapon or piece of armor is internally assigned by Blizzard and strongly correlates with the quantity of stats – such as strength or intelligence – with which the item imbues its bearer.

⁸Our estimates were invariant to aggregations of larger numbers of guild players, including measures based on the top 40 and top 50 contributors in the guild.

Table 29.1 Estimates for boss kill production

Variable	
α	0.342 (0.115)
PvP	-0.205 (0.078)
Implied efficiency	0.857

this is the result of the higher cost of obtaining raw materials, and/or the distractions of inter-player combat. It's interesting to note that within our sample, about 75% of guilds are on PvP servers despite the apparent disadvantage of doing so.⁹

Now we turn to the real meat of this estimation – the relative efficiency. As seen in Table 29.1, the average level of efficiency is 0.857. To reiterate, the relative efficiency measure is constructed to lie between 0 and 1, with 1 being the most efficient. This means that, amongst guilds in our sample, average efficiency is about 86% of the most efficient guild. The guilds in our sample are listed in Table 29.2, ranked according to their relative efficiency. For comparison, we include the number of boss kills for each guild. It seems to make sense that the highest ranked guilds are more efficient. However, we find that this relationship breaks down in the upper echelons of competition. In particular the highest ranked guilds are amongst the least efficient, while guilds somewhat below them in rank demonstrate greater efficiency. More interesting still is that the three highest ranked guilds in the US at the time of this analysis were called *Vodka*, *Blood Legion*, and *Premonition*, all three of whom ranked lower in efficiency than guilds who were slightly behind them in progression, such as *Vigil* and *Enigma*. Thus, if our hypothesized relationship between boss kills and fun is true, then our results further suggest that these highest-ranked guilds are not as good at producing that fun as slightly lower-ranked guilds.

29.4.1.3 Some Final Thoughts on SFA

These results are based on an analysis of a small test sample, and using data that were collected over a short period of time, so they should be read with a skeptical eye. However, the goal here was not to discuss model specification, rather, it was to provide a simple example of how SFA works and how it can be applied. As with any regression analysis, the results of SFA are sensitive to the specification and the sample selection, so some discussion of these issues is warranted.

Perhaps the most vital assumption that we have made is that we have controlled for all important sources of heterogeneity between guilds that could explain their rate of boss kills – namely, their raiding time, their armor and weapon quality, and

⁹ Indeed, this could be a point of criticism for our claim that guilds maximize fun only through boss kills. It looks as if, once a raiding dungeon's final boss is defeated, guild members like to have entertaining alternatives to raiding. Again, however, we appeal to our sample selection and to the time period under examination. Until the final boss is completed, boss kills are the only focus of the guild.

Table 29.2 Thirty competitive raiding guilds, ranked by boss killing efficiency

Guild name	Progression	Estimated efficiency
Enigma	6	0.945
Vigil	6	0.939
Exodus	6	0.937
TG	6	0.935
Temerity	4	0.927
Drow	5	0.922
No Chicks allowed	4	0.921
Void	4	0.911
Eternal Reign	5	0.906
Aptitude	4	0.904
Something Novel	4	0.900
Rebellion	4	0.898
Raiding Rainbows	4	0.896
Excessive Gaming	4	0.893
Incamate	4	0.885
Blood Legion	6	0.884
Pie Chart	4	0.879
Midwinter	4	0.875
Supermassive	4	0.868
Downtime	4	0.866
WHATEVER WERE AWESOME	4	0.862
TF	4	0.859
Blades of Wrath	3	0.852
Infallible	4	0.843
DotA AR BR Banlish ON	3	0.841
Reckoning	4	0.838
Gentlemens Club	4	0.833
Casual	3	0.824
Ropetown	4	0.820
Did it for Whitney	3	0.818
Coalition	3	0.817
FH	4	0.815
vodka	6	0.814
Predestined	3	0.812
Huge in Japan	3	0.806
Defenestrate	3	0.786
Premonition	6	0.777
Fallen	2	0.666
Astral	2	0.632

their server type. We also claimed, for the purpose of this exercise, that our sample of guilds was actually the full population of highly competitive US guilds. This is probably an unwarranted assertion. For example, it may be incorrect to lump the top two or three guilds in the US with their lower-ranked peers. The reason is that, whereas world top-ranked guilds are competing for world-first kills, lower-ranked

guilds may only be competing for region-first kills. To wit, best practices in tactics and strategy for guilds competing for different goals are likely to be different, so that their levels of efficiency are not strictly comparable.¹⁰ If we knew which guilds in our sample were competing for world-first kills and which were competing for region-first kills, we could modify our model accordingly. It may then happen that the top guilds are actually quite efficient when it comes to their particular kind of gaming goals. Similarly, if this estimation tool is applied to other player populations, be they in single or multiplayer games, it is extremely important that the analyst ensures that she has properly classified each player according to her goals.

We believe that SFA has many applications in both industry and in empirical academic work in game analytics. In industry, the general focus is on the analysis and improvement of game design with an eye to revenues, whether those are maximized through player retention, micropayments, or unit sales. One practical application of SFA for industry would be to gauge a game's success by considering how well the player interfaces with that game, and this is why we argue that it would be an excellent tool to evaluate the results of tinkering. A high variance of the efficiency term could suggest that some players have a very hard time learning how to enjoy the game under the introduced change. Additionally, it may be possible to use the estimated efficiency distribution to perform cluster analysis of players on the basis of the efficiency level predicted by a stochastic frontier model. To our knowledge, this has not been attempted.

Again, one of the most difficult questions to answer is what outcome players are seeking to maximize. We think it is reasonable to say that players are trying to have fun, but except for some very well-identified cases – like our guild example – it is hard to argue that fun can be captured by any one metric. Further exploration and research on this point is warranted.

For scholars in game analytics and other fields, the combination of games and SFA offers two fruitful avenues for research. For scholars who are interested in using game data to reveal how real life human institutions and organizations function, SFA is a great empirical tool because it supports more realistic models of behavior. For example, our work with guilds could also be considered to be a study in how well organizations complete multi-stage research and design processes, where each step depends on the completion of other tasks. This is the type of analysis for which SFA was first conceived is a natural avenue of further research.

SFA could also be useful to researchers who are specifically interested in gamer learning, or learning in general. Random assignment to treatment and control groups could easily be integrated into this model, but as far as we know, this has not been attempted. For researchers interested in human-computer interaction, SFA offers an interesting perspective in studies of how different interfaces affect perception of and completion of goals.

To sum up, we have introduced a way of measuring the effectiveness with which players make use of the game they are playing. We modeled player effectiveness as

¹⁰This is a hypothesis we will test in the next iteration of this study, for which the sample is much larger.

the efficiency with which a player approaches fun production, holding constant her available free time and the tools the game makes available to her to produce that fun. In the context of tinkering, SFA may offer developers and researchers a way to measure the response of players to mechanical or environmental changes, relative to the hypothesized, ideal approach. The greatest challenge, as with most analytical exercises, is deciding on a suitable set of proxies for fun. By example, we offer the meager insight that the value of a proxy may be decreasing in the number of people whose experiences it is used to evaluate. We have only given the barest of details about SFA, and encourage interested readers to consult one of the above-cited references for more expansive discussion.

29.4.2 Where Tinkering Is Needed

Up to this point, tinkering has been discussed as a phenomenon separate from social scientific theory. Tinkering game designers make discoveries not because they test theoretical predictions, but simply because they are fiddling around with tools. However, social scientific theory has a role in tinkering. It provides a basis for making predictions and understanding how something works. The SFA example above, for instance, used a production function that implied a theory about how players create fun. All game designers have theories about their player populations, but with few exceptions, designers do not develop them formally (Koster 2005). Formal theories of design are rare because the best designers generally do not have time, or because companies work to institutionalize knowledge that gives them a competitive edge. Yet, it seems that social scientists, game designers, and society as a whole could both benefit from a relationship where tinkering was parlayed into a more formal scientific process. By encouraging the development of formal theories for designing games that provide players with a sense of well-being, game designers can build better games and social scientists can use games to make a better society.

Take the current arrangement for loot distribution during pick-up raids in World of Warcraft. The system works under the standard need/greed/pass system that is common in MMOs. As a system, it works well for small groups and large groups of friends. However, under the current conditions – large semi-anonymous groups – players will “need” every item drop regardless of whether they can equip or use it. Due to this phenomenon, players who really need the item often have the probability of obtaining it drastically reduced. This is because there are a number of incentives – keeping items to trade them later, getting currency, grieving, or simply believing that other players are going to roll need – that drive players to behave selfishly and need all the time. This seemingly simple problem is known in game theory, and is referred to as a mixed-motive game. The group as a whole is better off if players only “need” items they can use and do not already own – this way the probability of getting a useful item is maximized. However, given that need always beats greed it only requires one cheater – or even the perception of a cheater – for a norm of “always need” to emerge. When players in a group recognize they are being cheated,

their only response is to “need” as well. Soon everyone in the group rolls need. Once those players move to a different raid, they carry the expectations of being cheated from the previous group and are more likely to “need”, which starts the cycle of “needing” in a new group. Hence, the behavior spreads through the server like a contagion.

The problem of loot distribution in raids has been solved in the past by guilds using DKP – Dragon Kill Points (Castronova and Fairfield 2007). However, the nature of pick up raids is that they are cross-server, one-shot interactions. Therefore, DKP will not work. In order to solve this type of problem designers could turn to social scientific theory, because identifying these types of situations is just the kind of work that social scientists have been doing for decades (Gintis 2000; Ostrom 2005; Schelling 1980).

29.4.3 The Social Influence of Norms

In 2006, Matthew Salganik, Peter Dodds, and Duncan Watts performed an experiment whose goal was to examine how individual behaviors lead to aggregate behaviors in a music market – a market that ranked music by the amount of times it was downloaded (Salganik et al. 2006). They called the experiment a multiple worlds experiment, because they performed experimental manipulations across multiple web servers. For each manipulation, they altered the number of times a song had been previously downloaded (social influence). The results that they encountered were not entirely clear. What they found was that two factors played a role in the eventual popularity of a song: song quality and number of previous downloads. What was interesting was that social influence had the strongest impact on songs that had an average quality rating. The best rose to the top and the worst fell to the bottom, but social influence determined the popularity of those in the middle.

Another interesting point is that a very weak manipulation of social influence (norms) had a strong impact on behavior. Especially where there was a degree of equivocality regarding the quality of the song. For game developers and social scientists, this presents an interesting result. This result, plus additional research, indicates that social norms act as powerful attractors and can synchronize the behavior of large groups of individuals. However, because social norms act as strong attractors they sometimes go haywire as seen in an information cascade or herding behavior (Anderson and Holt 1997; Watts 2002). When the normative choice is the wrong choice, a norm can lock a community in a sub-optimal social outcome, and become difficult for a community to escape its grasp. Because norms are strong attractors of human behavior it is important to understand how they come into being and if/how a social engineer or game designer could push a society to a better outcome.

Even with evidence that social norms can steer player behavior toward desirable outcomes it remains unclear just how game developers would accomplish such a task. Outside of community managers, social norms have not traditionally held

value for game designers as their emergent nature makes them difficult to understand and control. The process of actually harnessing social norms must be a twofold process. First, game designers should work with social scientists to familiarize themselves with existing theories, which explain how social norms and how human beings respond to them. Second, once developers have an understanding of the current state of the art, they can then apply and test it in their games. Of course, current understandings of norms are full of holes and anomalies. Game developers will have to perform controlled tinkering in order to determine where theory functions as expected and where it fails.

29.4.3.1 What Problems Can Norms Solve?

Imagine for a minute (avoiding the usual heated debate about the matter) that microtransactions are a good thing for both players and developers. To make this slightly easier, assume that in well-designed game microtransactions allow players to set their own personal price point. A player who wishes to pay hundreds of dollars a month for a game can do so, while a player who can only afford a few dollars a month can still enjoy the game. In doing so, microtransactions allow developers to capture two groups of players that they previously could not: players who cannot afford an entire subscription and players who are willing to pay more than the price of a subscription.

The monetization of free to play games using microtransactions has been a point of contention among game developers in recent years. The problem is that only a very small subset – unofficial reports indicate around 1–3% – of players actually pay to play the game. Increasing this number by a very small amount has the potential to lead to big increases in revenue. Developers have already been using tinkering to experiment with marketing and psychological theory as a means to increase player behavior (Ross 2011). For example, game analysts have performed A/B and multivariate testing to examine if anchoring – the tendency for an individual to attach himself or herself to a number they have recently seen when trying to make a numerical estimate – can increase the initial value assessment of virtual goods by players. However, perhaps using psychological tricks on players is the wrong approach. For many years, it has been normal for players to pay for games up front with no limitations on time or content. If players are not adopting microtransaction because of social norms, could developers push player behavior in a desired direction?

Another example where social norms could be valuable to designers is in creating online games with more meaningful social interactions. There are a number of situations where game designers would like to reduce antisocial behavior (e.g. griefing or trash talking) or increase prosocial behavior (e.g. mentoring or cooperation). In these situations, it can be difficult to sanction behavior because the behavior itself is difficult for the game designer to monitor. It is very difficult for a computer to differentiate between griefing and normal PVP. In addition, external sanctions may

lead to a decreased level of autonomy for players. Could norms solve this problem and push player behavior in the desired direction?

29.4.4 Harnessing Player Behavior

It is easy to suggest that norms can provide a solution to some of the behavioral problems that developers face, but it is more difficult to propose a solution. This is where tinkering and social science theory are useful. In order to solve these problems, designers must understand the motivations of players and the contexts in which norms become salient for those players. In addition, designers must also know whether they can shift or seed norms. Social science theory tells us that shifting norms can be difficult (Bicchieri 2006; Cialdini et al. 1990). Once a norm has become entrenched in a society, it is often very difficult for a society to move from one norm to another. However, there are cases when designers create the incentive structures and punishment mechanisms in ways that encourage a suboptimal social outcome. In these situations, it is important for game developers to understand social science theory.

Game developers have an interesting situation at their disposal. They have the opportunity to create more compelling game experiences while aiding the development of social scientific theory. Games are relatively novel environments and spaces for exploration and play. And even though players have the opportunity to explore, subtle cues in the form of narrative, mechanics or other players can push players out of their usual “real world” behaviors. What this suggests is that while players bring their own personal experiences and beliefs into the game environment, they also strive to learn the norms of the game environment in order to do well and fit in with the other players in the game.

Pre-established norms can also have an impact on behavior, because new players copy the behavior of others, and the social pressure from existing players encourages conformity (Asch 1956; Henrich and Boyd 1998). If players are actively searching and conforming to norms as they enter a game, then it should be possible for developers to establish preset norms desirable norms using beta communities. A developer could seed a world with a group hand-selected because they possess desired characteristics. A developer could hand-select a group of individuals who buy things to increase microtransactions, or individuals who punish antisocial behavior to decrease antisocial behavior. Still, putting players with desirable characteristics into a game will not be enough to change behavior. Norms are context sensitive so many elements of design influence how effective they can be. Designers must consider features, like communication channels, incentive structures, and group dynamics when trying to encourage desirable norms.

Norms are only one example of how social science theory can inform game design and controlled tinkering can feed back into social science theory. Yet, the game industry is a competitive arena, and even though controlled tinkering may have a positive impact on the bottom line, there is no guarantee. Tinkering with

games has vast potential for social science theory, and perhaps even the future of the human race. It is up to game developers to make the decision whether or not this type of investment in research and development is worth the cost.

29.5 Next Steps

Our brief summary of game analytics suggests that much advancement in social science will come from tinkering in games. Tinkering is a method that is ripe for discovery and social scientists should be observant of game designers tinkering with game societies. In addition, we presented a measurement tool for game industry tinkers – stochastic frontier analysis. In our example, SFA provides a way to operationalize engagement/fun for a segment of players who qualify as “Hard-core” raiders in World of Warcraft. However, SFA is not limited to the population in our example. It is our hope that developers use the method as a means to monitor engagement/fun for players with other interests. Finally, we presented an area of inquiry where game designers and social scientists might both benefit from careful tinkering – the use of social norms to shift player population.

We, thus, summarize the chapter takeaways to the following:

- The tool of the tinkerer is quite effective. In these areas of study (social norms, aggregate economic behavior, and so on), there’s never really been a tool like the large-scale game. No one has ever been able to split off groups of thousands of people and place them in exactly replicated incentive environments, then tweaked a thing or two to see what might happen. Moreover, this is happening at a time when our powers of observation are at their height. In online communities, very much can be observed.
- The questions of interest to the tinkerers are of interest to everyone. If a game company can figure out how to help its players live and play together happily, that knowledge will be of increasingly wide utility in an age when socializing is becoming ever more mediated.
- Stochastic Frontier Analysis is a tool that can help game developers operationalize engagement/fun for a population of similar players. Our example focused on “Hard-core” raiders in World of Warcraft, but SFA can be applied to players with different interests, in different games, as long as a metrics that are strongly correlated with fun can be determined. Designers and academics should try to combine survey data with measurements to ensure that they are measuring what they claim to be measuring by proxy.
- Increasing our understanding of social norms has positive consequences for both the real world (social science) and game design. One way that researchers and game designers can learn more about social norms is through controlled tinkering and research. However, if we are to increase this type of knowledge for the benefit of society game developers must share some of the burden in the risk and cost of controlled tinkering for the sake of discovering new knowledge.

About the Authors

Isaac Knowles is a PhD student in the Department of Telecommunications at Indiana University. His work focuses on the structure, regulation, and measurement of virtual economic activities and their relationship with the real economy. He received his BS in economics from Mary Washington College in Fredericksburg, Virginia in 2007 and his MS in economics from Louisiana State University in Baton Rouge in 2012. He has also worked as an economic research analyst at the US Federal Trade Commission.

Travis Ross is a PhD candidate in Telecommunications and Cognitive Science at Indiana University. His interests include the role of media in the formation of social norms, institutions, game mechanics, and using games as tools for social science experimentation. His recent research has investigated the role of cognitive heuristics in decision-making, and the influence of game design and social influence on individual attitudes, beliefs, and ultimately behavior.

Edward Castronova is a Professor of Telecommunications and Cognitive Science, Indiana University. Castronova (PhD Economics, Wisconsin, 1991) is a founder of scholarly online game studies and an expert on the societies of virtual worlds. Among his academic publications on these topics are two books: *Synthetic Worlds* (University of Chicago Press, 2005) and *Exodus to the Virtual World* (Palgrave, 2007). Professor Castronova teaches graduate and undergraduate courses on the design of games, the game industry, and the management of virtual societies. Outside his academic work, Professor Castronova makes regular appearances in mainstream media (60 min, the New York Times, and The Economist), gives keynotes at major conferences (Austin Game Conference, Digital Games Research Association Conference, Interactive Software Federation of Europe), and consults for business (McKinsey, Vivendi, Forrester).

References

- Aigner, D., Lovell, C. A. K., & Schmidt, P. (1977). Formulation and estimation of stochastic frontier production function models. *Journal of Econometrics*, 6(1), 21–37. doi:10.1016/0304-4076(77)90052-5.
- Anderson, L. R., & Holt, C. A. (1997). Information cascades in the laboratory. *The American Economic Review*, 87(5), 847–862.
- Asch, S. E. (1956). Studies of independence and conformity: A minority of one against a unanimous majority. *Psychological Monographs*, 70(9), 1–70.
- Axelrod, R., & Hamilton, W. (1981). The evolution of cooperation. *Science*, 211(4489), 1390.
- Bartle, R. (1996). Players who suit MUDs. *Journal of MUD Research*, 1(1). Retrieved March 5, 2012, from <http://www.brandeis.edu/pubs/jove/HTML/v1/bartle.html>.
- Beaman, L., Duflo, E., Pande, R., & Topalova, P. (2012). Female leadership raises aspirations and educational attainment for girls: A policy experiment in India. *Science*, 335(6068), 582–586. doi:10.1126/science.1212382.
- Bicchieri, C. (2006). *The grammar of society: The nature and dynamics of social norms*. New York: University of Cambridge Press.

- Castronova, E., & Fairfield, J. (2007). Dragon kill points: A summary whitepaper. SSRN eLibrary. Retrieved from <http://ssrn.com/paper=958945>.
- Cialdini, R. B., Reno, R. R., & Kallgren, C. A. (1990). A focus theory of normative conduct: Recycling the concept of norms to reduce littering in public places. *Journal of Personality and Social Psychology*, 58(6), 1015–1026. doi:10.1037/0022-3514.58.6.1015.
- Entertainment Software Association. (2011). Essential facts about the computer and video game industry. Retrieved March 14, 2012, from http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf.
- Gintis, H. (2000). *Game theory evolving*. Princeton: Princeton University Press.
- Greene, W. (2012). *Econometric analysis* (7th ed.). Boston: Prentice Hall.
- Henrich, J., & Boyd, R. (1998). The evolution of conformist transmission and the emergence of between-group differences. *Evolution and Human Behavior*, 19(4), 215–241. doi:10.1016/S1090-5138(98)00018-X.
- Koster, R. (2005). *A theory of fun for game design*. Scottsdale: Paraglyph Press.
- Kumbhakar, S., & Lovell, C. A. K. (2000). *Stochastic frontier analysis*. New York: Cambridge University Press.
- Lewis-Evans, B. (2012). Finding out what they think: A rough primer to user research, Part 2. Retrieved May 22, 2012, from http://www.gamasutra.com/view/feature/170332/finding_out_what_they_think_a_php.
- Mas-Collel, A., Michael, W., & Green, J. R. (1995). *Microeconomic theory*. New York: Oxford University Press.
- Mishra, S. K. (2007). A brief history of production functions. SSRN eLibrary. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1020577#.
- Ostrom, E. (2005). *Understanding institutional diversity*. Princeton: Princeton University Press.
- Rendell, L., Boyd, R., Cownden, D., Enquist, M., Eriksson, K., Feldman, M. W., Laland, K. N. (2010). Why copy others? Insights from the social learning strategies tournament. *Science*, 328(5975), 208–213. doi:10.1126/science.1184719.
- Ross, T. (2011). <http://www.motivateplay.com/2011/03/gdc-Tuesday-playdom-on-behavioral-economics-in-games/>. Motivate. Play. Retrieved from <http://www.motivateplay.com/2011/03/gdc-Tuesday-playdom-on-behavioral-economics-in-games/>.
- Salganik, M. J., Dodds, P. S., & Watts, D. J. (2006). Experimental study of inequality and unpredictability in an artificial cultural market. *Science*, 311(5762), 854–856. doi:10.1126/science.1121066.
- Schelling, T. (1980). *The strategy of conflict*. Cambridge, MA: Harvard University Press.
- Schelling, T. (2006). *Micromotives and macrobehavior*. New York: W.W. Norton & Company, Inc.
- Taylor, T. L. (2012). *Raising the stakes*. Cambridge, MA: The MIT Press.
- Watts, D. J. (2002). A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9), 5766–5771. doi:10.1073/pnas.082090499.
- Williams, D. (2010). The mapping principle, and a research framework for virtual worlds. *Communication Theory*, 20(4), 451–470. doi:10.1111/j.1468-2885.2010.01371.x.
- Witkowsky, E. (2012). *Inside the huddle: The sociology of team play in networked computer games*. PhD dissertation, IT University of Copenhagen.
- Yee, N. (2006). Motivations for play in online games. *Cyberpsychology & Behavior*, 9(6), 772–775. doi:10.1089/cpb.2006.9.772.

Chapter 30

Interview with Ola Holmdahl and Ivan Garde from Junebud

Alessandro Canossa

Junebud is a game development studio employing around 20 people and focused on digitally distributed, low threshold MMOs both for browsers and mobile phones. Junebud's business model revolves around digital distribution and free-to-play games, a couple of examples are: *Milmo* (Aeria Games, 2011) a browser game that blends 3D platform gaming with Action Adventure and RPG elements, and *Tuff Tanks* (Aeria Games, 2012) a turn-based action MMO for Ipad and Iphone.

Ola Holmdahl began in the 1990s as entrepreneur producing board and miniatures war games. He trained as illustrator and game designer and worked as lead game designer on *Battlefield 1942* (Electronic Arts, 2005) at DICE. He also received a master degree in sociology with a minor in philosophy of science. He also taught and was program manager for the game design line at the University of Skövde. In 2008 he founded Junebud where he is game designer and CEO.

Ivan Garde holds a bachelor in computer science from the University of Sao Paulo in Brazil. He has been working in the game industry as technical animator, game designer and product manager. Until recently, he worked at Mentez, the leading social game publisher in Latin America. In January 2012, he took on his current position at Junebud, where he now works as producer, business and metrics analyst.

Q: How do you think telemetry can be useful for the industry?

Ola: Junebud was started with the purpose to leverage digital distribution and freemium models. As we educated ourselves, we came to realize the crucial importance of accurate business intelligence and data about the user base. Game telemetry allows us to test hypotheses on game design right on the spot with players. When you design your

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

games as services, you are always updating them. Each update is a hypothesis, a question; we verify these hypotheses and answer questions with metric data: we discover what kind of engagement our users are looking for and what features are more likely to engage users. Similar approaches are used to maximize monetization. Junebud's products are online, permanent worlds, but I believe similar benefits can be had for single player, off-line games in terms of detecting hotspots, bottlenecks, confusing interfaces, weak story points, etc. The main problem is that once a problem is detected the game needs to be patched and that might not always be possible. Alternatively, it is possible to capitalize on the lessons learned and act on the next title released. But definitely these methods are more effective with shorter iterative cycles.

Q: Can you tell us what kinds of metrics do you track at Junebud?

Ivan: Our system tracks absolutely every player active on our servers, but we only monitor a subset of all actions available. The most basic set of metrics consists of log in and log out time in order to gain knowledge on how long and how often player access our games, but we are also interested in progression issues: what quests are initiated, when they are initiated and when they are concluded, if ever. Other progression systems that we monitor are the leveling systems (how quickly players level up) and spatial progression (where players tend to spend most time). We are also particularly aware of what items players have equipped and bought. We can also infer if an item is popular because of the price range or because of more intrinsic properties.

Q: Can you describe Junebud's telemetry setup?

Ola: We have deployed two parallel systems. *Milmo* adopts Overlord, a system that was developed in-house, whereas *Tuff Tanks* utilizes and off the shelf commercial solution: Mixpanel.¹

Q: Do you make use of telemetry data mostly to increase monetization or do you employ it also to perform user research?

Ola: We are interested equally in both and the two systems that we have in place tackle the issues in different manner. Overlord's dashboard categorizes the different data points that we track. The categories are: Quality of Service, Retention, Acquisition and Monetization. For each of these four macro categories we have a number of sub categories. For example, for Quality of Service the sub categories are: server up time, time to log in, average frames per second, average ping time, load time per level, etc. We care to constantly increase the Quality of Service to provide an attractive, trouble-free environment. Acquisition attempts to aggregate the virality of our applications and the sub categories are: how many invites are produced, how many invites turn into new players, etc. Overlord allows us to define benchmark values for each sub categories, a mark that we strive to hit. If the data collected is good enough or better than the established benchmark for all the sub categories, then the macro category is represented in green, while if any subcategory performs sub-par, the macro category is represented in red.

¹<https://mixpanel.com/>

Ivan: Regarding user research issues I want to tell you about our analytical setup for the first island in the game *Milmo*. We have broken down our database in terms of players that drop out in the first, second and third day, never to be seen again. We then analyze in detail what happened to those players: which quests did they engage with, which location did they spend most time, which level did they reach and which monster killed them prior to players quitting the game or whether players were still alive. We do this because we want to understand the difference in behavior between the period where players are allegedly having fun and the period just before quitting. In this manner, we were able to track a specific quest that caused several players to quit: it was confusing players and not allowing them to travel between the first island and the second. At this point we were able to iterate on the quest design and address the problems. The data we gathered was enough to form a hypothesis on the cause of frustration, that hypothesis proved correct when we started receiving data from the successive iteration: the problem did not appear again. Another example is the character customization screen; we created four versions of that and we tested all of them in parallel: a, b, c, and d testing, if you will. We were able to test which system produced the highest number of returning players increasing our retention of almost 2%.

Ola: In this case, we actually preferred investing in long-term retention. Some of the other options provided higher short-term retention rates but we chose the system that guaranteed highest long-term retention instead. *Milmo*, an MMO RPG adventure game, requires considerable time investment before it can shine to its full potential, hence we favored those players who showed the commitment to appreciate the game at its best. Also, we do not push monetization from the beginning; we let players choose further in the game whether they want to spend money.

Ivan: Other research practices involve mining player community through forums and traditional qualitative play testing. Although we did that a lot more at the beginning, before launch or just after, as it is now, our pipeline is not designed to combine the two different datasets. Recently, we discussed the benefits of setting up qualitative user tests.

Q: What analytical tools, in terms of algorithms, do you make use of?

Ivan: We tend to adopt simple techniques. We sample the data, look for averages and medians, basic descriptive statistics, but we don't do stochastic analyses. We often perform multivariate non-linear analyses, in my opinion single variable analyses are very prone to errors and false correlations (see Chap. 14). For example some time ago we were investigating the lowest acquisition cost per player and looking into different marketing models only to discover that players acquired with lowest spending were never monetized. We could discover that only because we broadened the view of our investigation. Simple correlations are enough to solve our problems; we rarely make use of advanced algorithms.

Ola: Another difference between Overlord and Mixpanel is precisely the power of the analytical tools offered. Mixpanel is stronger; we can create funnels, recurrence sheets and correlation analyses. With Overlord we often use spreadsheets to perform analyses.

Q: What are your practices in terms of visualization and reporting?

Ivan: It varies considerably; some data is presented as graphs, e.g., Average Revenue Per User (ARPU), Average Revenue Per Paying User (ARPPU) and regressions of ARPU//ARPPU over time. Regarding items popularity, we use layered bar charts that show us the differential between best and worst selling items. Retention and acquisition are also represented as overlaid graphs over time. As the associate producer, it is my duty to collect and collate the data several times a day. To generate the reports I use both spreadsheets and Overlord for *Milmo*, and Mixpanel alone is enough for *Tuff Tanks*. I then present the reports to Ola, the CEO, on a weekly basis. More often, when it's needed or an anomaly presents itself; I also present the material to the main stakeholders: the design and content production teams for each game. This practice has taken hold in all teams: they all look forward to the weekly meetings to evaluate their designs. The content production team finds it particularly beneficial to examine in detail the items charts; all new content is generated based on intelligence gathered from those reports. Designers focus more on player progression graphs over time and NPCs kill rates; these graphs can pinpoint design changes that affected player behavior, retention and ARPU/ARPPU. Even world and quest designers constantly make decisions supported by this information. Since we do not tend to embed precise location stamps to our data, we also do not resort to heatmaps, but our worlds are somewhat linear. Every 3 months an abstract overview master report is also presented to the board of directors.

Q: Can you describe a few success stories and not so successful stories on the use of telemetry at Junebud?

Ola: As mentioned earlier, the a/b testing we performed on the character editor allowed us to take informed, long term decisions on what kind of players we wanted to capitalize on. We also performed significant a/b tests on different intro sequences for *Milmo* trying to figure out whether to present the story first or start directly with basic interaction in the initial island and intersperse the story in smaller chunks. These a/b tests showcase in the best manner our metrics-based design processes.

Ivan: I'll open the box of mistakes that metrics analysis led us to; the most obvious case is the virality system that we created for Overlord. *Milmo* is the kind of game in which players are not required to involve their friends, a major difference from other games based on social networks. So we decided to include a progressive reward system that would give positive feedback for inviting an ever growing number of people. Although successful at the beginning, in the long run it faded and in a few weeks the virality score was back to where it started. We discovered that, unless a virality reinforcement system is integrated in the entire design loop of the game, it is not a viable strategy to graft one on an existing game. What happened is that for a short period a large number of non-committed players crowded and stressed the system, without increasing player experience.

Q: What are the next steps for Junebud?

Ivan: We want to keep improving the granularity of data we gather. We want to start looking at meaningful player metrics, such as deaths, and we want to include location stamps for all the events we track.

Ola: We also want to increase the number of variables we take into consideration for our multivariate analyses and step up the complexity of our algorithms with prediction and machine learning. In fact, purely descriptive statistics might lead us towards the fallacy of thinking that we know what our players are doing just because we track a large number of variables, which is why we want to push analyses over just reporting. In the future, I want us to start talking about what emotions our players are experiencing: frustration, joy, delight. I would like to connect the numbers we track with psychological profiles of players; if we don't do that the risk is to get blinded by data and mathematical probabilities.

Q: What advice would you give developers already working on MMOs and interested in implementing telemetry systems?

Ola: Make sure to win the development team over by showing the usefulness of metrics analysis for design. We had problems doing that initially, and unless everybody in the team is convinced of the benefits of metrics analysis, these are very difficult practices to enforce from above, even if we created the company around ideals of data-informed design. There was a lot of resistance, both from experienced professionals and people we recruited straight out of school.

Q: In which direction do you think it is possible to mature analysis practices in the industry?

Ola: By sharing! Maintaining silly confidential policies on analytical practices is counterproductive to the body of knowledge that we, as an industry, are trying to establish. Much knowledge is highly portable, between companies and between games. Clamping down on this knowledge will just trigger a big head-hunt season where the most experienced producers and analysts will be highly sought after and be paid a fortune, generating in the end a cast of techno-priests. And anyway, slowly, the experts are going to move from company to company, cross pollinating and eventually spreading the knowledge anyway. So, in the long run, it's a detrimental policy that, in order to maintain a fleeting edge over the competition, will waste time, increase costs and probably reduce the quality of games being produced.

Part VII

Metrics and Learning

Game developers developing games for entertainment mostly use telemetry analysis to gauge engagement and enjoyment. However, games for a purpose use metrics for a different purpose – analysis of impact of the game on the specific purpose. Thus, for games for learning the metrics obtained during game play can and should be used to assess learning and related learner variables.

This part of the book attempts to chart existing work on the subject by:

- Reviewing the literature of embedded assessment in games for learning.
- Constructing conceptually meaningful gameplay metrics to evaluate learning objectives.
- Considering how gameplay metrics can be utilized for adaptive learning environments.
- Presenting a series of relevant case studies.

The part consists of three chapters:

- Chapter 31: *Metrics in Simulations and Games for Learning*, a contribution by Jan Plass, Games for Learning Institute, NYU, Bruce D. Homer, CUNY Graduate Center, Charles K. Kinzer, Teachers College Columbia University, Yoo Kyung Chang, Teachers College Culmbia University, Jonathan Frye, G4LI, NYU, Walter Kaczetow, CUNY Graduate Center, Katherine Isbister and Ken Perlin from G4LI, NYU. This chapter explores the development of game metrics for learning. They present a case study to show examples of metrics applied in games developed at the G4LI.
- Chapter 32: *Conceptually Meaningful Metrics: Inferring Optimal Challenge and Mindset from Gameplay*, a contribution by Carrie Heeter and Yu-Hao Lee from Michigan State University, Ben Medler, and Brian Magerko from Georgia Tech University. The chapter explores the development of game metrics for games for learning. The chapter shows two case studies to show the approach in action.
- Chapter 33: *Interview with Simon Egenfeldt Nielsen from Serious Games Interactive*. This chapter is an interview with Simon Egenfeldt, both founder and managing director of Serious Game Interactive (SGI). The interview explores the use of game telemetry and analytics for serious games.

Chapter 31

Metrics in Simulations and Games for Learning

Jan L. Plass, Bruce D. Homer, Charles K. Kinzer, Yoo Kyung Chang,
Jonathan Frye, Walter Kaczetow, Katherine Isbister, and Ken Perlin

Take Away Points:

In this chapter, we will

1. Review the literature on embedded assessment in games for learning,
2. Introduce an approach to improving the diagnostic power of game metrics through learning mechanics and assessment mechanics,
3. Present a case study employing Cognitive Task Analysis for the study of learner behavior in science simulations,
4. Present two case studies illustrating the approach in learning games developed by G4LI.

31.1 Introduction

This chapter introduces the approach taken by the *Games for Learning Institute* (G4LI) to assess learning and related learner variables, with a focus on the use of metrics obtained during game play and simulation exploration. Learning is fundamental to all games (Gee 2008). At minimum, players must learn the basics of a game's mechanics to play. Additionally, players must uncover what these

J.L. Plass, Ph.D. (✉) • J. Frye • K. Isbister, Ph.D. • K. Perlin, Ph.D.
Games for Learning Institute (G4LI), New York University,
New York, NY, USA
e-mail: jan.plass@nyu.edu; jplass@gmail.com

B.D. Homer, Ph.D. • W. Kaczetow
Games for Learning Institute (G4LI), CUNY Graduate Center, New York, NY, USA

C.K. Kinzer, Ph.D. • Y.K. Chang, Ph.D.
Games for Learning Institute (G4LI), Teachers College Columbia University,
New York, NY, USA

mechanics are for, and what the game designer wants them to do (Cook 2006). Feedback mechanisms are an example of how game designers encourage (reward) or discourage (punish) a behavior. Game mechanics for learning must incorporate all of these aspects, from the moment-to-moment activities in which players engage, to reward and punishment systems.

In our research and game design work over the past several years we have found it helpful to use the following terms in the design of game mechanics (Plass et al. 2011):

Learning Mechanics describe essential high-level activities, grounded in learning sciences, that have learning as the primary objective;

Assessment Mechanics describe essential high-level activities, grounded in test theory, that have assessment as the primary objective;

Game Mechanics define the essential game play activity and can be based on learning mechanics, assessment mechanics, or both.

In our definitions, learning and assessment mechanics are meta-mechanics, i.e., descriptions of activities that become the foundation for corresponding game mechanics, following criteria to preserve their intended learning or assessment objective. That is, learning mechanics and assessment mechanics are constraints communicated to game designers who then select or design one or more corresponding game mechanics that accommodate these constraints.

Even though they do not use our terminology, commercial games make frequent use of our concepts of learning and assessment mechanics, though not to the extent required for games for learning. Learning mechanics are often used in commercial games for the tutorials rather than for conveying educational content. Let's use the example of *Mirror's Edge*, a first-person game that includes many complex maneuvers to traverse across rooftops and other urban terrain. The tutorial utilizes learning mechanics of *modeling*, which are grounded in the theory of Cognitive Apprenticeship (Collins 1988; Liu 1998). In addition to telling players what buttons to push, the game has a non-player character (NPC) model the action that needs to be performed. There are several game mechanics that are required to make this learning mechanic work, such as creating the NPC model's AI and positioning the camera for proper observation. All these game mechanics, as well as others, are implementations of a single learning mechanic.

Assessment mechanics in commercial games are often used in tutorials and throughout the game to ensure that players have learnt a certain feature and are using it correctly. In these kind of cases, game metrics are helpful in the design and playtesting process to assure that the experience is well designed, is at the correct difficulty, and is enjoyable overall. Other games, such as *America's Army* (AA) use metrics to rate players across several "army values"; including loyalty, duty, respect, selfless service, honor, integrity, and personal courage (America's Army 2012). Since these values are complex, assessment mechanics describe the player activities that can be used to assess the level of each of these values for any given player. These activities are then used by game designers to design the AA game mechanics. In the case of the complex AA values, each assessment mechanic was implemented through a combination of several game mechanics. For example, for a player to earn

points in loyalty, they must perform in-game actions such as mission objectives or neutralizing enemies while being connected to a teammate.

The purpose of using assessment mechanics in games for learning is to improve the diagnostic power of metrics collected during game play. Assessment mechanics help game designers select or design game mechanics that generate useful game metrics that allow us to measure variables related to learning, including learning outcomes (cognitive, behavioral, social, affective), trait variables, general state variables, and situation-specific state variables. In our approach, which we describe in this chapter, log data of game events and user behavior are also supplemented with observational data (video recordings or observer protocols) or eye gaze data that allow for triangulation of findings and therefore create more valid assessments of these variables of interest.

We believe there are two main benefits to our approach. First, embedded assessment, which uses measures based on metrics collected during game play, enables us to gain more detailed insights into learning than many traditional instruments allow with respect to the processes of learning and learning outcomes. This has implications for research as well as learner competency testing. Second, by using assessment mechanics to measure a series of learner variables, a learner model can be developed that allows us to design games that are individualized and adaptive to a learner's specific needs and characteristics. This has implications for the design of effective games for learning. In this chapter, we will discuss these implications and describe which variables to log, and which learner variables can be measured using game metrics. We will also provide illustrative case examples.

While most of this chapter concerns learning games and simulations, considerable progress has been made in using game metrics and telemetry in commercial games, examples of which have been discussed in this book. More examples include the Tracking Real-Time User Experience (TRUE) method (Kim et al. 2008) developed by Microsoft Game Studios, which combines several HCI approaches to track user data in a comprehensive system. The system tracks user actions and stores them in time-stamped log files that allow data to be analyzed as a sequence of events and "event sets", which adds contextual information that helps make sense of the event (Kim et al. 2008). Brief surveys are instrumented into games to collect attitudinal data, which allows direct user feedback. Data visualization within the TRUE system involves the creation of a meaningful hierarchical scheme and presenting the data that allows for navigating from high-level schemes to details and also linking between sources of information, such as log file data or video recordings of players (Kim et al. 2008).

Research has also looked at defining patterns of play using game metrics to identify player personas, which provide insight into the play-styles players consistently exhibit (Tychsen and Canossa 2008). Other research on commercial games has examined the use of time-spent reports (DeRosa 2007) and patterns of social behavior (Ducheneaut et al. 2004). Valve, the development studio behind *Half Life*, has also presented on the use of game metrics, survey data, and biometrics, such as skin conductance and eye tracking to gain a better understanding of players (Ambinder 2009). In addition, the many chapters of this book present several case studies of the use of

Table 31.1 Examples of variables that can be measured using game telemetry

General trait variables	General state variables	Situation-specific state variables
Executive functions	Achievement goal orientation	Emotional state
Spatial ability	Self-regulation, metacognition	Cognitive load
Verbal ability	Prior knowledge in the specific subject matter	Engagement
	Attitudes and interests	Situational interest

metrics (see Chaps. 4 and 8), their types and definitions (see Chap. 13), their visualizations (see Chaps. 18 and 19) as well as triangulation techniques (see Chaps. 21 and 22). In contrast, the work we are presenting in this chapter investigates variables and methods similar to those used by commercial game designers, but views usability, fun and engagement through the lens of learning game designers.

This chapter is intended for game designers interested in the design for games for learning, ideally with some background in the learning sciences. There are many questions related to game telemetry in practical applications that are beyond this chapter’s scope, such as how and when to introduce assessment tools in production, how to create a practical user interface to present the data, and how to prioritize assessment vs. content development given the constraints of finite resources.¹ Some of these issues, such as data reporting and visualization, are covered in Part IV of this book, other topics strictly pertaining to the educational domain will have to be addressed in other publications.

31.2 Review of Embedded Assessment in Games for Learning

The first two questions that must be considered when planning any assessment are: *what* is being assessed, and *why* is it being assessed? In other words, what variables do we want to measure, and what is the purpose of collecting this information? For educational materials, learning outcomes are the most common variables assessed to determine the effectiveness of an educational intervention. Learning outcomes are a way of conceptualizing the knowledge, skills, and abilities that the learner gains by playing the game. However, there are many other variables that may also be of interest. We have identified three broad categories of educationally relevant assessment variables: *General Trait Variables*, *General State Variables*, and *Situational-Specific Variables*, see Table 31.1.

Traits are relatively stable and are typically not targeted for change in educational games, but are often of interest and can be affected by game play (Anderson and Bavelier 2011). These include variables such as learners’ abilities (spatial, verbal, etc.)

¹ We thank Bill Shribman, one of our very thoughtful reviewers, for raising these questions.

and capabilities (e.g., executive functions). State variables, such as prior knowledge of subject matter, attitudes, self-regulation and meta-cognition, and goal orientation, are more malleable. They are therefore often the target of change in education games. Finally, situational variables are more transient and are often the result of learners' interactions with the game. They include emotional state, cognitive load, situational interest, and engagement. Effective educational materials, including learning games, are designed to influence situational variables in ways that maximize learning outcomes, for example, by reducing extraneous cognitive load and increasing engagement. By assessing these variables, one can determine if design choices have had the desired effect on relevant situational variables.

The purpose of assessment can be divided into two broad goals: description and evaluation:

Descriptive assessment provides a picture of the learners using an educational game. In general, descriptive assessment measures trait and demographic variables (such as age, gender, etc.) through questionnaires or surveys provided prior to engaging with educational materials.

Evaluative assessment attempts to measure something that is directly relevant to – and influenced by – the learning process. Evaluative assessment can be further broken into two types:

- *Formative assessment*, part of the instructional process; provides critical information to inform the learning process, and
- *Summative assessment*, typically conducted at the end of an educational intervention to determine if the intervention was successful. By combining descriptive and evaluative assessments, it is possible to determine if design choices are working for all learners.

Once it has been decided what will be measured and why, one must determine *how* to measure the variables of interest. In learning games, assessment can be embedded within the game's environment, making it unnecessary to employ survey and other paper-and-pencil measures for variables that can be measured based on the data collected by the game in user logs. Of course, there will always be variables requiring subjective observation or testing outside the game environment, perhaps to determine whether learners generalized the tasks taught by the game, or whether they can transfer knowledge to solve out-of-game problems. In the following section, we will discuss the ways in which this *embedded assessment* can be developed, and review some examples from the research literature.

31.2.1 Evidence Centered Design: Layers and Models

The most relied upon framework for developing embedded assessments in educational games is *Evidence Centered Design (ECD)* (Mislevy et al. 1999, 2002, 2003). ECD requires designers to be specific about what skills, knowledge, or other traits are being assessed, and what learners need to do, say, or create to provide evidence

relating to the variables being assessed. Although ECD can be applied to many assessment scenarios, it is particularly useful for guiding performance-based assessments, such as embedded assessments in learning games.

There are several layers of assessment design decisions in the ECD framework, including:

- *Domain Analysis* layer, where information is gathered about the domain of interest.
- *Domain Modeling* layer, where narrative assessment arguments are built, based on information gathered during domain analysis.
- *Conceptual Assessment Framework* layer, where narrative assessment arguments are translated into guidelines for specific tasks.
- *Assessment Implementation* layer, where assessment related tasks are presented to learners and responses are analyzed.
- *Assessment Delivery* layer, where learners' interactions are coordinated with assessment tasks and assessments are scored and reported.

31.2.2 ECD Assessment: A Closer Look

ECD assessment layers are guided by the *conceptual assessment framework*, consisting of three models in which relevant activities take place. The first is *Student Models*, which identifies relevant *skills, knowledge, identity, values, and epistemology* within a specific learning domain (Shaffer 2006). The student model ranges from simple, even including only one variable, to complex, involving multivariate item response theory or latent class modeling (Mislevy and Riconscente 2005). *Evidence Models* describe what learners must say, do, create, etc. to provide evidence related to the skills, knowledge, etc. being assessed. They also provide scoring rules and statistical models for how observed data relate to components of the student model. *Task Models* specify how variables from the evidence model will be collected in the educational game. What will learners say, do or create to provide evidence of the skills, knowledge, etc., being assessed? ECD also includes a *presentation model* that describes how relevant materials are presented to learners throughout the assessment process. Although the different ECD models can be considered separately, their interconnectedness means that effective assessment requires them to be considered jointly and revised iteratively (Rupp et al. 2010).

A number of the assessment approaches used in educational games are either derived directly from an ECD approach, or fit easily into the ECD framework. For example, Shute and her colleagues (Shute et al. 2009) used ECD to seamlessly embed assessments into educational games in an approach they call *stealth assessment*, because the assessment is not obvious to the learner.

With stealth assessment, Bayesian networks are used in the evidence model to determine how performance in the educational game relates to specific variables in

Model-Based Assessment: Examples

Shute and her colleagues have conducted a number of studies using ECD and Bayesian networks to embed a stealth assessment into commercial and education video games. Shute et al. (2009) assessed creative problem solving in *Oblivion*, a commercial roleplaying game where players encounter a river they need to cross that was full of dangerous fish. The ways that players solved this river-crossing task was evaluated to assess players' creative problem solving skills. Two *Oblivion* experts rated possible solutions with respect to *novelty* and *efficiency*, two concepts identified as being important for creative problem solving.

The concept model of creative problem solving identified two main sub components, problem solving and creativity (Shute et al. 2009). Both have elements of efficiency, which Shute et al. defined as the quality and quantity of the steps taken to reach a solution. Novelty, defined as choosing low frequency solutions, was considered independent of problem solving. The evidence model for this assessment had to define the connections between observable behaviors, creativity and problem solving. The action model defined interactions between the student and the game that were used to glean data. Each action taken to solve a problem (recorded in log file metrics) was scored on novelty and efficiency. Bayesian networks were used to covertly assess the river-crossing task. This approach has the potential to analyze tasks in games that allow experts to define the concept model and devise scoring rules (Shute et al. 2009; Shute 2011), with the possible exception of emergent gameplay where all possible solutions cannot be known initially and evaluated.

the competency model (Shute and Kim 2011). Bayesian Networks are directed graphs with probabilities attached to each node (Mislevy and Gitomer 1996). Each node corresponds to a variable of interest from the student model. The first step in implementing a Bayesian network is to define the structure of the relevant content domain (Mislevy and Gitomer 1996). Next the structural relationships must be transformed from relationships between constructs to relationships between probability distributions (as part of the conceptual assessment framework). These probability distributions are computed using empirical evidence and/or theory. Distributions of variables can then be obtained based on other variables that are at a level above the variable of interest. The probability of a given event is calculated recursively, as data is fed into the system (Mislevy and Gitomer 1996). A dynamic Bayesian network functions in a similar way, but also considers time as a variable (Iseli et al. 2010).

While the use of Bayesian networks is part of ECD's conceptual assessment framework, *Cognitive Task Analysis* (Schraagen et al. 2000; Williamson et al. 2004)

is used primarily to support the domain modeling layer. CTA identifies the knowledge representation and cognitive processes that underlie complex behaviors, such as job performance or problem solving (Williamson et al. 2004).

Although the specific details of how CTA is carried out can vary, it generally begins with the identification of the tasks required to perform a target action or achieve a learning goal. The skills and knowledge necessary to accomplish these tasks are then identified, most often by developing a visual representation (Lajoie et al. 1998; Roth and Woods 1989). Once the tasks and the underlying knowledge representation are identified, a variety of techniques are used to have learners verbalize their cognitive and metacognitive processes as they perform the tasks (Cordingley 1989; Roth and Woods 1989). The information provided during the vocalizations is then used to inform the design of assessment tools. While many CTAs are completed manually, there are approaches to automate them (Shute and Torrance 2003). This ECD approach has been used to create embedded assessment in a number of educational games. However, there are other approaches that did not explicitly use the ECD framework, such as described by Heeter et al. (2009).

31.3 Case Study: Assessment of Meta-cognition in Simulations Using User Logs

Although Bayesian networks are particularly useful when the variables of interest are complex and their behavioral expressions are not well defined, Bayesian networks are not always needed, particularly for variables with clearly identified behaviors. This is the situation with our first case study, which utilized the *Behavioral Measure of Metacognitive Processes* (BMMP), a method to assess learners' metacognitive processes from metrics data. We developed this approach, a modified version of CTA, to examine students' metacognition as they interacted with science simulations (Chang et al. 2008; Chang and Plass 2012). An advantage of conducting assessment with metrics data is that it can provide causal explanations of why and how learning was effective (Winne 1982) by capturing the process through which learners demonstrate their development of specific competencies (Nelson et al. 2010).

Metacognition comprises higher order thinking, reasoning, and learning skills (McCombs 2001); this process is employed by learners to direct their learning processes. Metacognition affects learning by influencing how students use resources in the learning environment (Aleven and Koedinger 2000; Hill and Hannafin 2001), and by mediating the appropriate use and regulation of cognitive resources (Schraw and Moshman 1995).

31.3.1 Description of the BMMP

The BMMP is a framework for the construction of a rubric to identify metacognitive processes from metrics data, i.e., from user behaviors captured in log files (Chang et al. 2008; Chang and Plass 2012). The resulting BMMP rubric is a set of behavioral patterns that manifest metacognitive control within the specific learning environment. The BMMP can be applied to different student-directed exploratory learning environments, including games, simulations, and microworlds.

The large quantity of metrics data poses challenges to data analysis. By employing Cognitive Tasks Analysis (CTA), the BMMP can address the limitations associated with using metrics data, namely the difficulty of observing internal processes through manifest forms, and the selection bias in the variables being observed (Efklides 2002; Perry et al. 2002). CTA is able to identify the cognitive structures and processes beyond the observable behavior (Brown 1997), provide rich descriptions of implicit psychological processes and reduce selection bias by ensuring that all relevant cognitive strategies are included.

We established the validity of the BMMP in a study in which BMMP data was triangulated with think-aloud data. In addition, our research has shown that the BMMP was able to reliably assess learners' metacognitive processes across different learning environments (Chang et al. 2008; Chang and Plass 2012).

BMMP begins by identifying the learning goals addressed in the learning environment. Then, a CTA is conducted to develop a rubric, identifying the behavioral evidence of metacognitive control by determining: (1) the tasks required to accomplish the learning goal, (2) the skills and knowledge necessary to accomplish these tasks, and (3) the observation of participants on task, often accompanied by interviews, concurrent verbalization, or verbal prompts. The BMMP adopts the CTA's first two phases to identify the tasks and metacognitive strategies required to accomplish the learning goal. The result is a set of behavioral patterns, or a coding rubric, which can identify metacognitive processes from the metrics data.

31.3.1.1 Case Example—Assessment of Metacognitive Processes with Interactive Computer Simulations

We applied the BMMP to assess high school students' ($N = 457$) metacognitive processes while learning with two interactive computer simulations for chemistry, one on Kinetic Molecular Theory and the other on the Ideal Gas Laws. The relationship between students' prior knowledge, metacognitive processes, and the learning outcome, especially in comparison to the self-report measure of students' regulation of their learning processes, were examined. Both simulations involve a set of variables related to molecular movement. Understanding the relationships between the variables is essential for understanding the chemistry concepts. With the simulations, students can explore the chemistry phenomena by: (1) setting the hypothesis about the relationship between the variables, (2) experimenting with the variables by

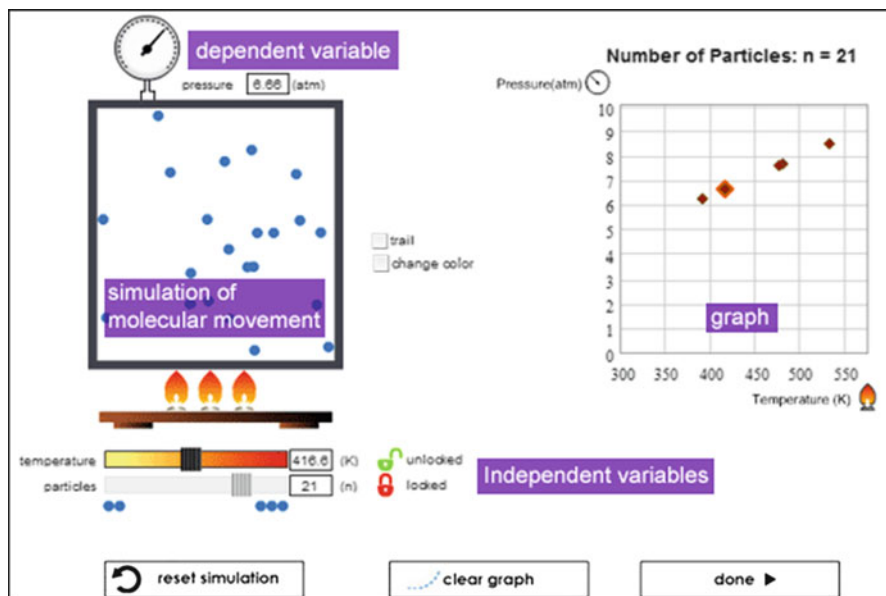


Fig. 31.1 Interactive computer simulation for kinetic molecular theory

directly changing the value of independent variables, and (3) observing the results through a visual simulation of molecular movement and a graph that displays the results of the experiment (Fig. 31.1).

We assessed students' *prior knowledge* using a knowledge pre-test, and *Self-Regulation* using a self-report measure, the *Motivated Strategies for Learning Questionnaire* (MSLQ; Pintrich et al. 1993). The BMMP assessed the quality of metacognitive control based on the metrics data from simulations. Finally, learning outcomes were measured for comprehension and transfer of knowledge.

Development of the BMMP Rubric

Step 1: Analysis of the Learning Goals. First, a content analysis for each of the interactive computer simulations was conducted to identify learning goals. Both simulations' learning goals consist of understanding the relationships between independent variables (e.g., temperature) and dependent variables (e.g., internal pressure, rate of diffusion) (see Table 31.2).

Step 2: Cognitive Task Analysis (CTA). A Cognitive Task Analysis identified the behavioral representation of metacognitive control and the conditions under which such patterns can lead to the achievement of the learning goals. The result of the CTA was a rubric containing behavioral patterns that manifest metacognitive control in a given learning environment.

Table 31.2 Learning Goals for M&M Simulations

Simulation	Learning goals
Ideal gas laws	Internal pressure is directly proportional to temperature Internal pressure is inversely proportional to volume Volume is directly proportional to temperature
Kinetic molecular theory	Internal pressure is directly proportional to temperature Internal pressure is directly proportional to number of particles Matter (gases) are made up of particles in constant random motion

Table 31.3 BMMP rubric for the first learning goal of Ideal gas laws simulation

Behavior pattern	Condition	Strategy	Observation
Temperature (T) – T ...	Two or more consecutive manipulations	Experimentation and observation of relationship between T and P while Volume (V) is locked (held constant)	T increase, P increase T decrease, P decrease
Pressure (P) – P ...	Two or more consecutive manipulations	Experimentation and observation of relationship between P and T	P increase, T increase P decrease, T decrease

With our simulations, the learning goals consist of relationships among pairs of variables (i.e., one independent and one dependent variable). The first learning goal of the Ideal Gas Laws simulation was to understand that *internal pressure is directly proportional to temperature*. At least two consecutive observations are needed to identify a rule defining the relationship between each pair of variables (Gick and Holyoak 1983; Schwartz and Black 1990). Therefore, student behavior of manipulating the temperature variable two or more times consecutively while holding the volume variable constant was recognized as reflecting a metacognitive strategy for this learning goal (see Table 31.3).

Application of the BMMP Rubric. The resulting BMMP rubric was applied to code metrics data and compute a BMMP measure. BMMP measures for the simulations were computed as a sum of two proportions to reflect the two types of metacognitive control required to accomplish the learning goals of the simulations: *adaptive behaviors*, i.e., the number of adaptive behaviors as identified by the BMMP rubric over total behaviors, and the *number of adaptively explored variables* over all variables available for exploration.

For both simulations, prior knowledge and metacognitive processes evaluated via the BMMP predicted learning outcomes. Interestingly, the self-report measure of self-regulation did not predict learning, suggesting that the BMMP-based behavioral measure is better able to capture learning relevant metacognitive processes than a paper-based survey of self-reports; see Chang (2010) for details.

This case example shows that CTA is a feasible way to establish a task model for measuring metacognition. The resulting BMMP measure was able to predict learning

in two different simulations, whereas a self-report measure of metacognition was not. In comparison to open-ended sandbox games, the simulations used for this study are relatively limited in their exploration options. It is therefore reasonable to expect that an application of this approach to an environment that provides more opportunities for self-regulation will yield a highly valid and reliable way of measuring learners' metacognition using game metrics.

31.4 G4LI Approach to Assessment in Games for Learning

In this section, we will discuss the approach for assessment of learning and related learner variables taken by the Games for Learning Institute (G4LI). We will first define game mechanics in general, and then introduce the concepts of learning mechanics and assessment mechanics. We next describe how our approach extends the ECD model by adding observational data to the task model. We also provide case examples of how this learning and assessment approach was applied in two games, and summarize the studies we conducted with these games to identify initial design patterns for games for learning.

31.4.1 *Game Mechanics*

The concept of *game mechanics* is central to understanding games. Game studies scholars and developers have offered definitions that range from finely detailed to broad and conceptual approaches. In general, game mechanics are “the various actions, behaviors and control mechanisms afforded to the player within a game context” (Hunicke et al. 2004, p. 3). In a narrower sense, a game’s core mechanic is “the essential play activity players perform again and again and again” (Salen and Zimmerman 2003, p. 316). In other words, a game’s core mechanic contains the moment-to-moment actions and interactions in which the player engages while playing the game. Finally, a definition proposed by Cook (2006) describes game mechanics as “rule based system/simulations that facilitate and encourage a user to explore and learn the properties of their possibility space through the use of feedback mechanisms” (p. 1). Game mechanics are, therefore, a means to guide players into particular behaviors by constraining the space of possible plans to attain goals (Järvinen 2008). Game mechanics do not only apply to players, as they are also defined as “methods invoked by agents, designed for interaction with the game state” (Sicart 2008), a definition that includes the game AI as actor, in addition to the player.

31.4.2 *Learning Mechanics*

When games are designed with the explicit goal of facilitating learning, game mechanics must go beyond making a game fun and engaging—they must engage players in meaningful learning activities. The game mechanic becomes an integral part of the

learning activity. Game designers have long seen this connection and have argued that new mechanics are needed to engage the player in a way that facilitates learning (Isbister et al. 2010). Most importantly, the designers interviewed by Isbister et al. made a strong case that learning needs to be embedded in a game's core mechanics rather than added on to existing mechanics. Game play cannot be used as a reward for answering questions; and vice versa, questions cannot be forced into unrelated game play.

An example of poor integration of learning into a game is when a learning game uses an established game mechanic, such as a shooter mechanic, and the learning mechanic consists of a popup question that must be answered before players can resume the game.

To emphasize this qualitative difference of mechanics that are designed for learning, we offer the following definition of what we call *learning mechanics*:

Learning mechanics are patterns of behavior or building blocks of learner interactivity, which may be a single action or a set of interrelated actions that form the essential learning activity that is repeated throughout a game.

Learning mechanics adapt the moment-to-moment activity of a game mechanic into a meaningful learning activity, often expressed as player choice. All games offer players series of choices and then react to those choices with new challenges. Learning mechanics can push game mechanics further to offer players choices that help them learn as well as facilitate gameplay. A game's learning aspects become an integral part of the game play rather than an addendum to the game mechanic.

The relationship between game mechanics and learning mechanics is that learning mechanics are meta-mechanics that become the foundation for the design of a corresponding game mechanic or collection of game mechanics. Learning mechanics are not themselves playable mechanics. They describe the functions of the actions available to players in the game, but they don't describe the actions themselves. For example, the learning mechanic might specify that the learner/player should be able to apply rules to solve problems, but does not describe the corresponding game mechanic. Concretely speaking, in the game *Angry Birds* there are three core gameplay mechanics:

- the slingshot system,
- the simulated laws of gravity affecting birds, pigs and other pieces,
- the physical static simulation applied to all non player-controlled elements consisting of mass, structural solidity and friction.

The learning mechanic in this case could be based on a constructivist approach, or simply a “learn by doing” principle:

- match properties of different objects (here, different types of birds: black/bomb, yellow/acceleration, green/Boomerang, etc.) to properties of the different materials (here, materials of structures protecting the pigs: glass, wood, stone),
- explore the rules that bind trajectory, velocity and mass.

In this *Angry Birds* example, these two learning mechanics are instantiated through the combination of the three core mechanics described above. In the context of another game, however, the same learning mechanics could be instantiated by using other game elements and mechanics, such as different types of air planes for which players draw the flight path, such as in *Flight Control*, or using jet packs to guide your character to a specific place, such as in *Little Big Planet*.

Learning mechanics are meta-mechanics that can be formulated independent of a specific game (though often linked to a specific game genre), and that need to be instantiated as game mechanics to describe the concrete actions and their affordances for the players in the system that the game represents. Playing the game equals learning these tools and moves, becoming familiar with them, and having the satisfaction of solving challenges, of “beating” the game (Juul 2003). Additionally, in designing game mechanics based in learning mechanics, game designers consider the game feel, the feel of engaging the core mechanics through interactive elements, visual elements, emotional elements and sound elements (Swink 2008).

31.4.2.1 Criteria for Effective Learning Mechanics

Several criteria can ensure that Learning Mechanics will be effective in facilitating learning. Most importantly, learning mechanics are grounded in learning sciences and learning theory; many such theories and frameworks have been developed that can be used as the basis for the design of learning mechanics. Examples include *Cognitive Flexibility Theory* (Spiro et al. 1988; Spiro and Jehng 1990), *Cognitive Apprenticeship* (Collins 1988; Liu 1998), *Anchored instruction* (CTGV 1990, 1993), and *Situated Learning* (Lave and Wenger 1990). From these and other, related theories, designers choose activities that engage the learner in meaningful interaction with a specific subject.

The interactions in learning mechanics should be designed from a clear model of interactivity types. For example, the INTERACT model distinguishes three types of interactivity: behavioral, cognitive, and emotional interactivity, and describes their relation in learning processes, such as feedback and guidance (Domagk et al. 2010). Learning designers use these interactions to describe actions that allow learners to generate solutions to the problems that are designed to facilitate learning. Models like INTERACT help designers identify what behaviors (e.g., clicks, controller button actions, or touch actions) and emotion to engage the learner to facilitate thinking and decision making. This is a noteworthy difference to definitions of game mechanics, which question a deterministic relation between inputs, controls and mechanics (Järvinen 2008; Sicart 2008).

31.4.2.2 Requirements for Designing Game Mechanics Based on Learning Mechanics

In the creative process of instantiating learning mechanics as game mechanics, game designers must ensure that the learning goal is preserved. Thus, when selecting

Table 31.4 Library of learning mechanics and associated game mechanics

Learning mechanic	Corresponding game mechanics
Reciprocal teaching: learner teaches target concepts to NPCs	G4LI <i>Noobs v. Leets</i> reciprocal teaching
Learner places icons representing key concepts to solve problems	G4LI <i>Supertransformations!</i> Problem solving using reflection and rotation icons
Learner creates authentic problems to solve by other players	G4LI AR Simulation Game for Science Learning mechanic of creating locations where specific scientific principles were applied

a game mechanic to implement a particular learning mechanic, designers need to consider several requirements, including:

1. *Don't introduce excessive demands by unrelated information that distract from the learning goal.* For example, when the goal is to solve a science problem, the game narrative should not present excessive amounts of unrelated information.
2. *Don't introduce excessive demands by unrelated tasks that distract from the learning goal.* For example, when the goal is to collect essential information in the game, the collection task should not be so difficult that the learner cannot move on.
3. *Keep it challenging—don't excessively reduce the amount of the required mental effort to process the game's essential learning content.* For example, when solving algebraic equations, the game should not automatically indicate the changed sign of a term that is being subtracted.

A more detailed discussion of these requirements is beyond the scope of this chapter, but appears in Plass et al. (2011c).

31.4.2.3 Library of Learning Mechanics

To provide learning game designers with a set of learning mechanics and associated instances of game mechanics that are useful for their game designs, we have begun to compile a library of mechanics. This library includes a variety of game mechanics options for each learning mechanic; see Table 31.4 for examples and <http://g4li.org> for updates.

Note that since learning mechanics are meta-mechanics, there is a one-to-many relationship of learning mechanics to corresponding game mechanics, and each of the different game mechanics that instantiates a Learning Mechanic may only be suitable under specific conditions—for specific learners, topics, or game genres. Our ongoing work is concerned with adding new learning mechanics and associated game mechanics, and with demonstrating their viability and usefulness through empirical research.

31.4.3 *Assessment Mechanics*

In addition to engaging learners in meaningful activities that facilitate learning and assist in the creation of mental models, games have the ability to provide educators and designers, as well as players, with insight into players' learning processes and advancements. The rule systems created by game mechanics can be used to assess a variety of variables, including learning outcomes.

Learning objectives and learning processes of interest can be operationalized into specific actions within the game in ways that allow us to assess achievement levels. Player actions can be captured within the game telemetry, as log files, and can be analyzed to reveal what, and how, players have learnt. For example, different problems in the game might each contribute to a score describing the player's mastery of a particular sub-skill. Game mechanics for assessment must therefore be designed to elicit relevant behaviors that can be observed through the user log and interpreted to reveal learning process, outcome, and learner variables. We call mechanics designed for this purpose *assessment mechanics*.

Assessment Mechanics are patterns of behavior or building blocks of diagnostic interactivity, which may be a single action or a set of interrelated actions that form the essential diagnostic activity that is repeated throughout a game.

Similar to learning mechanics, assessment mechanics are meta-mechanics that describe player actions but are not playable mechanics. They describe the functions of the actions available to players to demonstrate their knowledge, skills or expressions of other variables of interest, but they don't describe the assessment tasks themselves. For example, the assessment mechanic might specify that the game should engage the players in activities in which they group related items in time or space, but does not describe the corresponding game mechanics, e.g., whether the grouping is done by shifting items on the screen like in *Bejeweled*, dropping them in specific locations like in *Drop Seven*, or placing them like in a *tower defense* game.

In our example of *Angry Birds* used in the Learning Mechanics section, the assessment mechanics may describe a set of actions that allow us to determine to what extent players have learnt to match properties of the different types of birds (black, green, blue) to properties of the different materials (glass, wood, stone). In this particular case, succeeding in demolishing a structure with a single shot can be interpreted as evidence for a high level of achievement of this learning goal. However, this performance is also indication that players have a good understanding of what trajectory and which force need to be applied to a certain bird, which confounds conclusions about individual outcomes, such as learning to match birds and materials. The assessment mechanic would therefore ask for the implementation of multiple levels with varying problems that allow us to disentangle the different player competencies.

Several variables are of interest in designing personalized or adaptive games (see Table 31.1, above). As discussed above, these variables can be grouped as general trait, general state, and situation-specific state variables. Some of these variables can be reliably assessed with valid traditional instruments, but for many variables

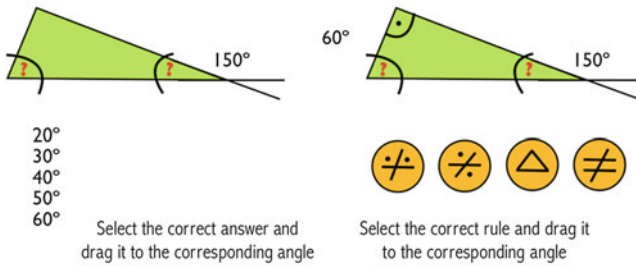


Fig. 31.2 Two assessment mechanic options in Noobs vs. Leets

only methods with low reliability and validity are available, often involving self-reported data, which are susceptible to learner biases and other response sets. Each variable shown in Table 31.1 is important because it predicts learning and is therefore useful to be measured in-game.

For example, learners' self-regulation is of interest, because it describes whether players establish learning goals, monitor their goal achievement, and change strategies when they are unable to achieve their goals. Our first case example (above) showed how we measured self-regulation in science simulations based on a Cognitive Task Analysis.

Another example is the assessment of specific aspects of learning. In the game *Noobs vs. Leets* (see our third case example below), we were interested in understanding how well the learner comprehends rules related to angles in triangles and quadrilaterals, such as the complementary angles, and opposite angles rules. We, therefore, chose an assessment mechanic requiring learners to drag the correct rule to the angle to be solved (Fig. 31.2, right). An alternative assessment mechanic, requiring the learner to drag or enter the correct numeric value for each angle (Fig. 31.2, left) could not have revealed the source of possible errors, which could have been conceptual (lack of rule knowledge) or arithmetic (lack of subtraction skills). This reasoning is based on the Evidence-Centered Design (ECD) Framework (Mislevy et al. 2003).

This framework, described in more detail above, provides a formal approach to the essential questions related to assessment design: what should be assessed; what kinds of learner behaviors can be used to reveal these constructs, and, what tasks and activities can be designed to elicit these behaviors. Below we will use the ECD model to develop criteria that mechanics must meet to be useful assessment mechanics.

31.4.3.1 Criteria for Assessment Mechanics

Similar to learning mechanics, assessment mechanics must meet criteria to assure they engage the player in meaningful and valid assessment activities. The goal of assessment mechanics is to elicit relevant behaviors that can be observed through the user log and interpreted to reveal learning processes, learning outcomes, and

learner variables. To be useful for this purpose, i.e., to make metrics a valid measure producing reliable scores, the first criterion is that assessment mechanics must be based on assessment models, such as ECD. Based on the student model of target competencies, described in relation to gains in skills, knowledge, identity, values, and epistemology that students are supposed to obtain (Rupp et al. 2010), assessment designers need to construct an ECD Evidence Model. Thus, they must specify the salient features of learner behavior and the rules for scoring and interpreting these features for assessment purposes. This involves another important criterion for designing assessment mechanics: the consideration of test-theoretical concerns. For example, since we cannot assume that individual test items are independent of one another, the assessment's statistical model must reflect these possible dependencies (Rupp et al. 2010).

Another criterion is that the mechanics need to be designed based on aspects of the ECD Action Model, i.e., the description of the key tasks and activities in which the learner will engage within the assessment's purpose. These tasks form the assessment mechanic, and it is essential that they be designed so that the task execution can be captured through the game's instrumentation. This means, for example, that the mechanics require the learner to make explicit the steps used for problem solving rather than simply provide the answer to the problem. The mechanics need to create repeated exposures to similar problems to allow for multiple observations of the target behavior.

Depending on the designers' decisions, the character of the assessment mechanic may or may not be obvious to learner. We describe assessments in which learners are aware that they are being assessed as *embedded assessment*, and those where they are unaware of this fact as *stealth assessment* (Shute 2011).

Once measurement experts have designed an assessment mechanic, game designers can design corresponding instances of game mechanics. However, in this process, several design requirements must be met.

31.4.3.2 Requirements for Designing Game Mechanics Based on Assessment Mechanics

One of the most common problems in designing game mechanics based on assessment mechanics is the introduction of confounds that make it difficult to determine whether variability in learning scores among learners can be attributed to their different knowledge and skills or to other factors. One such confound is the addition of new information to be processed and tasks to be executed. For example, game mechanics, such as in *Flight Control*, where players have to determine the approach patterns of airplanes for landing, are fun and engaging because the fast succession of a high number of planes to land puts high demands on players' processing. This could appropriately assess speed of processing, but not conceptual knowledge or higher level thinking. A related confound is the addition of scaffolding or guidance that reduces cognitive task demands for some learners, but not for others. For example, if key information in an adventure game is hidden,

Table 31.5 Library of assessment mechanics and associated game mechanics

Assessment mechanic	Corresponding game mechanics
Learners apply rules to solve problems	Fling mechanic in <i>Angry Birds</i> Drag mechanic in <i>Explode!</i> Rule mechanic in <i>Noobs v. Leets</i>
Learners arrange items in time or space to solve problems	Drag mechanic in <i>Gravity</i> Flight path mechanic in <i>Flight Control</i> Tile placement mechanic in <i>Plumber!Toobz</i>
Learners select items that belong to each other in time or space to solve problems	Drag mechanic in <i>Osmosis</i> Selection mechanic in <i>Bejeweled</i>

resulting in only some players finding it, then assessment of knowledge will be confounded by learners' exploration strategies and player type.

Another typical problem for assessment is that the game mechanic introduces confounds through demands on fine motor skills, which are highly variable in learners. For example, *MotionMath* asks learners to tilt their tablet device to direct a ball to the correct answer. Success in this task not only depends on learners' knowledge, but also on their ability to move the ball to the correct location.

Likewise, many game mechanics include activities that require learners to have content knowledge or skills from unrelated subject matter areas. For example, an assessment of algebra may be confounded by the need to know about Newtonian physics. Although integrating different subject matter areas is a desirable design feature for learning mechanics, doing so in designing assessment mechanics may confound results.

A final confounding variable to consider is emotion. During game play, learners will likely experience a series of emotional responses that would impact learning outcomes (Um et al. 2011). Designers of assessment mechanics must consider learners' emotions and design mechanics that are aimed at assessment so that the learners' emotional response does not interfere with their ability to solve the problems presented. A particularly problematic situation would result from mechanics in which different people emotionally respond in different ways.

31.4.3.3 Library of Assessment Mechanics

To provide learning game designers with a set of assessment mechanics and associated game mechanics that they can use for their game designs, we have begun to compile a library of mechanics. This library lists a variety of game mechanics options for each assessment mechanic, see Table 31.5 for examples and <http://g4li.org> for updates.

Note that there is a one-to-many relationship of assessment mechanics to game mechanics, and that each of the different game mechanics that can instantiate an Assessment Mechanic may only be suitable under specific conditions. Our ongoing work is concerned with adding new assessment mechanics and associated game mechanics, and with demonstrating their viability and usefulness through empirical research.

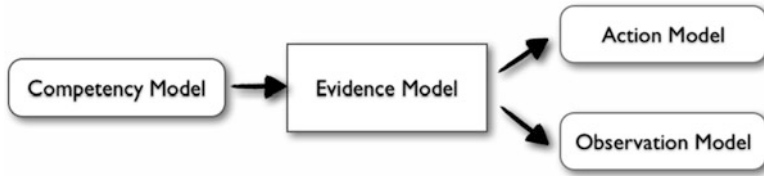


Fig. 31.3 Expanded ECD framework with added observational model

31.4.4 Expanding the ECD Model to Include Observational Measures

The current ECD model describes how the implementation and analysis of game metrics is based on a systematic process of describing the competencies to be acquired by learners in a competency model. It is crucial to specify the evidence that needs to be collected to be able to assess the degree of competency achieved by a student, and then define an action model that describes learners' actions, characteristics and contextual information. We have extended this framework to allow for additional evidence that goes beyond this strictly behavioral model by incorporating observational measures (see Fig. 31.3).

Observational measures are operationally defined as measures allowing observation of the learner through a variety of measures, including video observations, classroom observers, eye tracking, and biometric sensors such as EKG, GSR, EMG, and EEG.

Adding observational measures to the ECD model expands the evidence model to include new forms of evidence to describe a specific competency or learner variable. Using observational data and data provided by game metrics as evidence of a specific target performance allows triangulation of findings to enhance measurement validity (see Part V “Mixed Methods for Game Evaluation” and Chap. 26). In fact, many of the variables of interest discussed above (see Table 31.1) have correlates in physiological responses. For example, learners' cognitive load can be assessed through behavioral measures of problem solving or other learning tasks, but also through measures of pupil dilation, heart rate, galvanic skin response, and EEG (Isbister and Schaffer 2008). In the following section, we will describe two case examples of how we use this expanded ECD model in our research on games and learning.

31.5 Case Example: *FactorReactor*

FactorReactor is a G4LI-developed game that focuses on improving math fluency and cognitive flexibility for number sense. Players must complete a series of math problems, but they have a choice of what operations they perform, and on what numbers.

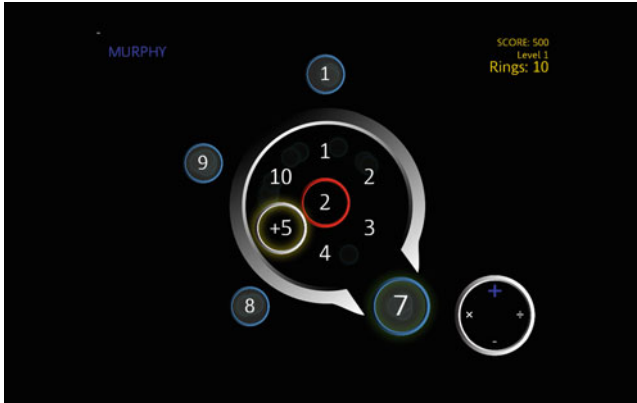


Fig. 31.4 Screen shot of the G4LI Game *FactorReactor*

31.5.1 Game Description

The Learning Mechanic in *FactorReactor* involves choosing mathematical operations and a second number to change the player's number to the goal number.

The game mechanic instantiating this learning mechanic places the number that the player is trying to manipulate at the center of the screen (Fig. 31.4). The numbers adjacent to the center number and inside the silver circle are the numbers that players can use to change their center number. The numbers outside the silver ring are the players' goals. The object of the game is to make the center number equal each of the goal numbers in the outermost circle.

Players use the Xbox 360's analog stick controller to move the selector circle to the number that they want to use. Mathematical operations (plus, minus, divide, multiply) are mapped to the controller's colored buttons. To perform an operation players pull the right trigger, and to swap the selected number with the center number players pull the left trigger. Also, players can change what goal number they are attempting by pressing the left and right shoulder buttons (Fig. 31.5).

Feedback in the game consists of a point system and a resource system. For the point system, players are rewarded with points for every goal they solve. The resource system is conveyed as rings. Each operation players performs costs 1 ring, and they are rewarded with 5 rings for every goal that is solved. Learners choose what mathematical operations to perform and in what order to implement them. Additionally, since each operation uses one ring, players must be efficient in solving problems occasionally using division and multiplication, or they will run out of rings and have to restart the entire level.

FactorReactor's game mechanics are simple, to reduce players' cognitive load and instantiate the learning mechanic without adding excessive amounts of extraneous load. This allows players to devote more resources to learning activities instead of memorizing what buttons they should press or how to perform an action in the game. Other options, perhaps involving animals or magic spells, could have been

Fig. 31.5 Mapping of controls for the G4LI Game *FactorReactor*



used, but math fluency is about speed and accuracy, so creating an easy to learn control scheme and simple representations allow players to quickly move from one math operation to the next.

31.5.2 Assessment Mechanics

As discussed above, assessment mechanics describe building blocks of diagnostic activity. In many games, the game mechanics used to instantiate the assessment mechanic are the same as the game mechanics used to instantiate learning mechanics. *FactorReactor* is an example of such a game. The goal of the assessment was to first determine players' initial ability level with mathematical operations, and then to measure how players improved their use of those operations and the speed at which they solved equations.

We conducted a study with *FactorReactor* to determine which mode of play was most effective for increasing math fluency. Sixty-four 6th to 8th grade students were assigned to three versions of the game: *solo*, *collaborative*, and *competitive* (Plass et al. [in press](#)). In the solo condition, each student played the game by him or herself. In the collaborative and competitive game versions, students played in pairs, either helping each other or trying to beat each other's score. In preparation, participants first watched a tutorial video and were given a short practice session to become adjusted to the gameplay. In addition to playing the game, participants were asked to complete surveys on situational interest and achievement goal orientation, and pre/post math fluency tests. They also completed play sessions as pre- and post tests. Game play metrics were saved as log files.

Table 31.6 Selected log data fields captured in *Factor Reactor* game

<i>General information</i>
User ID
Game version
Treatment
<i>Learning and performance indicators</i>
Total # of levels completed
Total # of problems solved
Total # of unique problems solved
Total # of rings
Total score
Total # of “+” operations used per solution
Total # of “-” operations used per solution
Total # of “x” operations used per solution
Total # of “÷” operations used per solution
Total # of operators used per solution
<i>Self-Regulation Patterns</i>
Total # of game overs
Total # of operations used per solution
Total # of operations used per game over
<i>Cognitive Flexibility</i>
Total # of “swap” operations used
Total # of “x” operations used per solution
Total # of “÷” operations used per solution

31.5.3 Game Telemetry

FactorReactor was instrumented to capture in-game player behavior in a log file. The assessment mechanic was designed to allow us to capture the specific variables within the game that would provide evidence of what rules players were selecting, how players implemented the rules and in what order, and if the players were successful. Perhaps the simplest question to answer with the telemetry data was what operators players were using. We stored each player’s action, which allowed us to count all additions, subtractions, multiplications, and divisions players performed on each attempt at a level. Thus, if players failed and restarted a level, we could distinguish between what they did on their first attempt and how their strategy may have changed with each subsequent attempt (Table 31.6).

The frequency of each operation provides insight into players’ ability level and their comfort with each operation. For instance, our study of middle school students revealed that many participants made infrequent use of multiplication and division, preferring to stick with simpler operations of addition and subtraction.

Other telemetry data that were useful in understanding player performance included time taken to solve a goal and to solve a level, number of attempts per

level, and rings used per attempt. For example, an analysis of the overall number of problems solved and controlling covariates, such as pre-test game performance and degree of experience with a video game controller, found that players in the competitive condition performed significantly better than players in the individual condition, but not significantly better than those in the collaborative condition.

The metric of time taken can offer a direct assessment of ability level in situations where the demands of processing unrelated information and executing unrelated tasks is minimized. For instance, in cases where control schemes are difficult to master or confusing, time taken may only reflect the players' struggles to input their knowledge. This can be assessed through the use of pre- and post-tests of the educational content delivered in a more traditional way, such as a paper and pencil assessment. Time-stamped user actions can also be analyzed in other ways. For instance, we can mark the time each operation was used in the log files to determine what order players used for operations when solving a goal or level. This can indicate players' strategies, such as using the common concept of order of operations.

We did not assess players' cognitive flexibility with a paper-based measure, but the user logs allowed us to analyze what operations were used, how many swaps were performed, how often the player changed goals, and the time taken to solve a goal. Each of these variables can be analyzed based on each attempt to solve a problem, level, or overall. Players with higher cognitive flexibility for the task are likely to solve problems faster, use more diverse operations, and to use swapping and goal switching.

31.5.4 Future Directions

In addition to the metrics discussed, psychophysiological measures can provide rich sources of information on players. With *FactorReactor*, we have begun collecting data from participants using an eyetracker. Eyetracking data allows us to mark areas of interest on the gameplay screen and analyze a variety of different variables related to the players' gaze. For example, we can measure the frequency that players look at a specific area of the screen, or how long they spend fixated on that area. We can also analyze areas of interest that players commonly transition between. In our *FactorReactor* study, for example, we can compare the competitive condition with the other conditions and determine if players spent more time or glanced more frequently at their score or the score of their opponents. Eyetracking data can provide information about player strategy, player motivation, and resource management. Eyetracking can also provide data regarding extraneous cognitive load, if it shows that players frequently glance at areas of the screen containing information that is not useful for game play.

31.6 Case Example: *Noobs vs. Leets*

Our final case example describes *Noobs vs. Leets*, another game developed at G4LI, designed to teach middle school geometry. More specifically, players learn about different types of angles and rules that can be used to determine missing angles in quadrilaterals.

31.6.1 Game Description

Noobs vs. Leets uses a simple narrative in which players must guide their characters, noobs, across levels to save their fellow noobs from their dreaded enemy, the leets. Through identifying angles, players open up pathways that allow them to reach their goal. The game is divided into six chapters, each with about eight levels. In each chapter students learn about a different angle type.

Chapter 1 starts with simple angles. As players progress, they encounter complementary angles, supplementary angles, vertical angles, triangles, and finally quadrilaterals. The beginning of each chapter includes a cut-scene that introduces the new angle type.

Figure 31.6 provides an example of a typical level midway through the game. Each level starts with the noob parachuting in from an airplane and starting at the

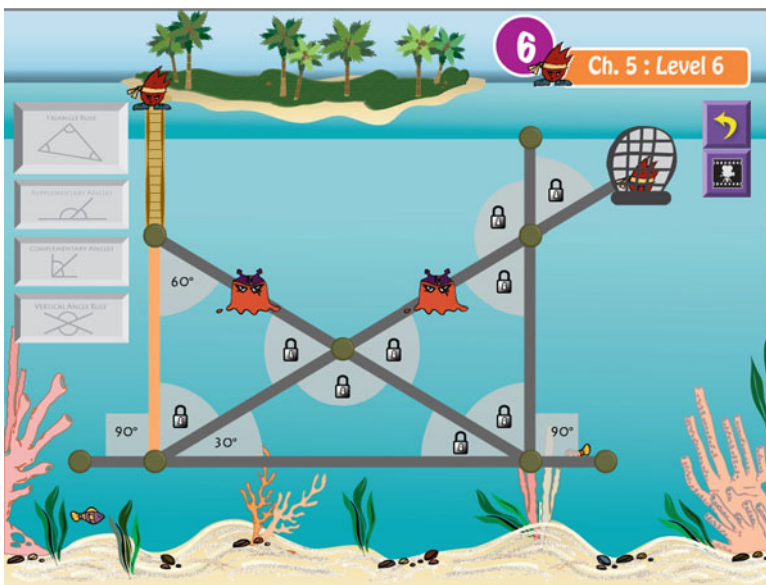


Fig. 31.6 Chapter 5 of the G4LI Game *Noobs v. Leets*

upper left portion of the screen. By using known angles, players can solve locked angles by applying the correct rule. On the left side of the screen are four buttons, corresponding to different types of angles. As players progress through the game, they learn about and apply new and more complex rules. Each time a player unlocks an angle, a new pathway opens. The goal is for players to reach the noob trapped in the cage in order to set it free. On some levels, there are stationary or mobile leets that try to knock the player off the level and force them to restart the level from the beginning. Each time a level is solved, the player earns another noob; however, if they choose the wrong angle or run into a leet, they lose a noob. The number of noobs a player has saved is indicated next to the chapter/level indicator. On the right side of the screen, players have access to a button that allows them to move to a previous level or re-watch the cut-scene for that chapter.

31.6.2 *Learning Mechanics*

Noobs vs. Leets provides a useful example of how learning mechanics can shape a game mechanic. The learning mechanic is *Apply Rules to solve Problems: Learner selects among different rules and indicates for which problems they apply*.

Although *FactorReactor* and *Noobs vs. Leets* look quite different and have dissimilar play, there are similarities in their underlying learning mechanics. In both games, players have to choose the rule (angles or mathematical operations) and apply it where they see fit. One difference is that rules are of a more conceptual level in *Noobs vs. Leets*. To assist learning, the first few levels of each chapter provide scaffolding to establish a foundational knowledge for that new rule. As the player progresses through each chapter, levels shift from practice and scaffolds to problem solving. Players must decide not only what rule is most applicable, but also what route is the best to take through the level.

31.6.3 *Assessment Mechanics*

The assessment mechanics for *Noobs vs. Leets* was designed to allow us to decide if players understood the rules they were applying, how successful they were at implementing the rules, and if they were able to improve as they progressed in the game. For example, we were interested in gathering evidence about players' ability to grasp the concept of complementary angles and whether they were able to apply it with fewer mistakes as they finished each level.

We conducted a study with 89, 6th to 8th grade participants to assess if it was more important for players to apply the rule to an angle (as described above) or if they would learn more from calculating the angle (Plass et al. 2012). For this purpose we created two versions of *Noobs vs. Leets*. One worked as described above; the second version included number buttons instead of rule buttons and players had

Table 31.7 Selected log data fields captured in *Noobs v. Leets* game

General information
User ID
Game version
Treatment
Learning and performance indicators
Total # of levels completed
Total # of unique correct solutions
Total # of unique correct solutions per chapter
Total # of incorrect solutions
Total # of incorrect solutions per chapter
Total play time
Time per chapter
Total # of noobs lost
Total # of noobs lost per chapter
Total “Game over” screens
Total clicks on vertices
Total clicks on angles
Total # of unique correct uses for each rule
Total # of incorrect uses for each rule
Time on each tutorial cut-scenes
Total time on tutorial cut-scenes
Total number of times player returned to the tutorial cut-scene for each chapter
Average reaction time to level per chapter
Total # of deaths to leets
Self-regulation patterns
Average # of consecutive incorrect rule applications in a row using the same rule
Longest string of consecutive incorrect uses of each rule
Cognitive flexibility
Average number of different rules used per level
Highest number of different rules used on a level

to calculate the locked angles and then click on the correct number of degrees for that angle rather than selecting the rule that they would apply. The difference between the two treatments, therefore, was that one explicitly captured which rule players applied, whereas the other only captured the numeric responses for solved angles. Participants played the game for 25 min and completed a situational interest survey, a math fluency test, and a pretest/posttest of geometry knowledge. The math fluency test and pretest allowed us to run analyses controlling for prior knowledge.

31.6.4 Game Telemetry

For this study, we were interested in capturing telemetry data in the log files that would allow us to compare our two conditions in several ways. Table 31.7

summarizes variables that were tracked in the log files. These variables were assessed to measure general information, learning and performance indicators, self-regulation, and cognitive flexibility.

Telemetry data paired with surveys and pre- and posttest measures allowed us to analyze the data from several perspectives. For example, results indicated that although players in the number condition had greater situational interest in the game, players in the rules condition completed many more levels of the game. As for learning outcomes, we found that players who used the rule-based game continued increasing their knowledge as they completed more levels, whereas players of the number-based version did not increase their knowledge past a certain level. Results of players in the conceptual or rule condition did not exhibit an interaction between test performance and levels completed. For a more detailed analysis and additional results, see Plass et al. (2012).

31.6.5 Future Directions

In our next iteration of research using *Noobs v. Leets*, we are incorporating an incentive system that uses badges as rewards for different accomplishments in the game. The badges are designed based on goal orientation theory, with some badges appealing to a mastery orientation, while others appeal to a performance orientation (Ames and Archer 1988; Elliot 2005). For this study, we will be instrumenting the log files to show how many badges players earned and how often they clicked to view their badges. In addition, we will use eyetracking data to determine if there is some optimal level of badge implementation. Are there cases where players look at badges too much and become distracted from the learning objectives? Or do the badges provide sufficient motivation to improve learning?

Additionally, work is underway examining how the two versions of the game, the conceptual and the numeric, interact with student engagement and prior knowledge. Preliminary results suggest students with higher ability beliefs in math enjoy solving problems numerically, whereas students with lower ability beliefs tend to prefer the rule based system. Such work has implications for how learning mechanics are integrated into games for learning. Work in this area is also examining motivation over time within the game, and the effect of varying choice and feedback within assessment mechanics.

31.7 Summary and Conclusion

This chapter introduced the approach for assessment of learning and related learner variables taken by the Games for Learning Institute (G4LI), with a particular focus on the use of metrics obtained during game play. Our approach suggests that game mechanics, the essential game play activity, should be distinguished from learning mechanics and assessment mechanics. We define learning mechanics as essential

high-level activities, grounded in learning sciences; they have learning as the primary objective. In contrast, assessment mechanics are essential high-level activities, grounded in test theory, that have assessment as the primary objective. Learning and assessment mechanics are meta-mechanics that can be instantiated as corresponding game mechanics, following criteria we outlined above to preserve their intended teaching or assessment objective.

Variables related to learning that can be measured through game metrics include learning outcomes (cognitive and skills), trait variables, general state variables, and situation-specific state variables. Supplementing log data of game events and user behavior with observational data extends the ECD model and can result in more valid assessments of these variables.

Our approach serves two related but separate goals. One is a measurement goal—embedded assessment allows for more detailed insights into learning than many traditional instruments, both with respect to the process of learning and learning outcomes. This has implications for research as well as learner competency testing. The other goal is related to improving game play. By using assessment mechanics to measure a series of learner variables, a competency model can be compiled, allowing for the design of games that are individualized and adaptive to a learner's specific needs and characteristics. This has implications for the design of effective games for learning by making games more adaptive and personalized, and, hopefully, more effective.

Using this approach, designers of games for learning have a strategy at their disposal that allows them to design the mechanics of their game based on the specific goal of either learning or assessment of learning, or both. The resulting learning and assessment mechanics communicate these functions to game designers and provide corresponding constraints for the design of game mechanics. For each game and game genre these mechanics will differ, and libraries of mechanics, such as the ones under development by G4LI, will provide useful resources for learning game designers.

In summary, we described an approach that, grounded in theory and tested in several game design projects, has implications both for research and practice of the design of games for learning. While many other approaches have applied learning theory and assessment theory to game design, the approach presented here provides a practical way for teams designing games for learning to separate the role of learning scientists (designers of learning mechanics), assessment experts (designers of assessment mechanics), and game designers (designers of the corresponding game mechanics).

About the Authors

Jan L. Plass, Ph.D. is Paulette Goddard Professor of Digital Media and Learning Sciences at NYU Steinhardt. He is the founding director of the Consortium for Research and Evaluation of Advanced Technology in Education (CREATE), and co-director of the Games for Learning Institute (G4LI).

Bruce D. Homer, Ph.D. is an Associate Professor of Educational Psychology (Learning, Development & Instruction Sub-Program) and Training Director for the

Interdisciplinary Postdoctoral Research Training program at the Graduate Center at the City University of New York (CUNY).

Charles K. Kinzer, Ph.D. is a Professor of Communication and Education at Teachers College Columbia University where he is the program coordinator at Computing, Communication and Technology in Education (CCTE).

Yoo Kyung Chang, Ph.D. is a Lecturer in CCTE at Teachers College Columbia University and former research assistant at the CREATE lab at New York University.

Jonathan Frye is the Technology Coordinator and a Research Assistant for the CREATE lab. He is currently a doctoral candidate in the Educational Communications and Technology program at New York University.

Walter Kaczetow is currently a doctoral student in CUNY Graduate Center's program in Educational Psychology. When not studying Walter can be found teaching mathematics as an adjunct professor at New Jersey City University.

Katherine Isbister, Ph.D. is an Associate Professor of both Computer Science Engineering and Digital Media at NYU's Polytechnic Institute. She is the founding director of the Social Game Lab.

Ken Perlin, Ph.D. is a professor in the Department of Computer Science at New York University. In addition to being the director of the Games For Learning Institute, he was also founding director of the Media Research Laboratory and director of the NYU Center for Advanced Technology.

Recommended Readings

- Iseli, M. R., Koenig, A. D., Lee, J. J., & Wainess, R. (2010). *Automatic assessment of complex task performance in games and simulations* (CRESST Report 775). Los Angeles: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).
- Kim, J. H., Gunn, D. V., Schuh, E., Phillips, B., Pagulayan, R. J., & Wixon, D. (2008). Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (pp. 443–452). Florence: ACM. doi:[10.1145/1357054.1357126](https://doi.org/10.1145/1357054.1357126)
- Plass, J. L., O'Keefe, P., Homer, B. D., Case, J., Hayward, E. O., Stein, M., & Perlin, K. (in press). Motivational and cognitive outcomes associated with individual, competitive, and collaborative game play. Special issue on advanced learning technologies. *Journal of Educational Psychology*.
- Plass, J. L., Homer, B. D., Kinzer, C., Frye, J., & Perlin, K. (2011, September 30). *Learning mechanics and assessment mechanics for games for learning* (G4LI White Paper # 01/2011 Version 0.1). Available online at www.g4li.org
- Shute, V. J., Ventura, M., Bauer, M. I., & Zapata-Rivera, D. (2009). Melding the power of serious games and embedded assessment to monitor and foster learning: Flow and grow. In U. Ritterfeld, M. Cody, & P. Vorderer (Eds.), *Serious games: Mechanisms and effects* (pp. 295–321). Mahwah: Routledge/Taylor & Francis.

References

- Aleven, V., & Koedinger, K. R. (2000). Limitations of student control: Do students know when they need help? In Gauthier, G., Frasson, C., & VanLehn, K. (Eds.), *Proceedings of the 5th international conference on intelligent tutoring systems, ITS 2000* (pp. 292–303). Berlin: Springer.
- Ambinder, M. (2009). *Valve's approach to playtesting: The application of empiricism*. Presented at the 2009 Game Developers Conference, San Francisco.
- America's Army. (2012). *America's Army: Honor and advances – Army values*. Retrieved July 3, 2012 from http://manual.americasarmy.com/index.php/Honor_and_Advancement
- Ames, C., & Archer, J. (1988). Achievement goals in the classroom: Students' learning strategies and motivation processes. *Journal of Educational Psychology, 80*(3), 260–267.
- Anderson, A., & Bavelier, D. (2011). Action game play as a tool to enhance perception, attention and cognition. In S. Tobias & J. D. Fleycher (Eds.), *Computer games and instruction* (pp. 307–330). Charlotte: Information Age Publishing: IAP.
- Brown, T. L. (1997). *Task analysis strategies and practices* (Practices Application Brief). Washington, DC: Office of Educational Research and Improvement. (ERIC Document Reproduction Service No. ED 404 571).
- Chang, Y. K. (2010). *Examining metacognitive processes in exploratory computer-based learning environments using activity log analysis*. Unpublished Doctoral Dissertation, New York University, New York.
- Chang, Y. K., & Plass, J. L. (2012). Assessment of the metacognitive processes from the behavioral data. (G4LI White Paper # 01/2012). Available online at www.g4li.org
- Chang, Y. K., Plass, J. L., & Homer, B. D. (2008, October). *Development and validation of a behavioral measure of metacognitive processes (BMMP)*. Featured research presentation at the annual convention of the Association for Educational Communication and Technology (AECT), Orlando, FL.
- Cognition and Technology Group at Vanderbilt. (1990). Anchored instruction and its relationship to situated cognition. *Educational Researcher, 19*, 2–10.
- Cognition and Technology Group at Vanderbilt. (1993). Anchored instruction and situated cognition revisited. *Educational Technology, 33*(3), 52–70.
- Collins, A. (1988). *Cognitive apprenticeship and instructional technology*. Hillsdale: Lawrence Erlbaum.
- Cook, D. (2006). What are game mechanics? *lostgarden.com*. Retrieved May 23, 2010 from <http://lostgarden.com/2006/10/what-are-game-mechanics.html>
- Cordingley, E. S. (1989). Knowledge elicitation techniques for knowledge-based systems. In D. Diaper (Ed.), *Knowledge elicitation: Principles, techniques, and application* (pp. 89–175). New York: Wiley.
- DeRosa, P. (2007, August 7). Tracking player feedback to improve gamedesign. *Gamasutra*.
- Domagk, S., Schwartz, R., & Plass, J. L. (2010). Defining interactivity in multimedia learning. *Computers in Human Behavior, 26*, 1024–1033. doi:10.1016/j.chb.2010.03.003.
- Ducheneaut, N., Moore, R. J., & Nickell, E. (2004). Designing for sociability in massively multiplayer games: An examination of the “third places” of SWG. In J. H. Smith & M. Sicart (Eds.), *Proceedings of the Other Players Conference*. Copenhagen, Denmark.
- Efklides, A. (2002). The systemic nature of metacognitive experiences: Feelings, judgment, and their interrelation. In M. Izaute, P. Chambres, & P.-J. Marescaux (Eds.), *Metacognition: Process, function, and use* (pp. 19–34). Dordrecht: Kluwer.
- Elliot, A. J. (2005). A conceptual history of the achievement goal construct. In A. J. Elliot & C. S. Dweck (Eds.), *Handbook of competence and motivation* (pp. 52–72). New York: Guilford Publications.
- Gee, J. P. (2008). *What video games have to teach us about learning and literacy* (Rev. and updated). Basingstoke: Palgrave Macmillan.
- Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15*, 1–38.

- Heeter, C., Magerko, B., Medler, B., & Fitzgerald, J. (2009). Game design and the challenge-avoiding 'Validator' player type. *International Journal of Gaming and Computer-Mediated Simulations*, 1(3), 53–67.
- Hill, J. R., & Hannafin, M. J. (2001). Teaching and learning in digital environments: The resurgence of resource-based learning. *Educational Technology Research and Development*, 49, 15–26.
- Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A formal approach to game design and game research. In *Proceedings of the Challenges in Game AI Workshop, 19th National Conference on Artificial Intelligence, AAAI'04*, San Jose, CA. Vancouver: AAAI Press.
- Isbister, K., & Schaffer, N. (2008). *Game usability. Advice from the experts for advancing the payer experience*. New York: Morgan Kaufman.
- Isbister, K., Flanagan, M., & Hash, C. (2010). Designing games for learning: Insights from conversations with designers. In *Proceedings of CHI (Conference on human factors in computing)*, Atlanta, GA.
- Järvinen, A. (2008). *Games without frontiers: Theories and methods for game studies and design*. Tampere: Tampere University Press.
- Juul, J. (2003). The game, the player, the world: Looking for a heart of gameness. In M. Copier & J. Raessens (Eds.), *Level up: Digital games research conference proceedings* (pp. 30–45). Utrecht: Utrecht University.
- Lajoie, S. P., Azevedo, R., & Fleiszer, D. M. (1998). Cognitive tools for assessment and learning in a high flow information environment. *Journal of Educational Computing Research*, 18(3), 205–235.
- Lave, J., & Wenger, E. (1990). *Situated learning: Legitimate peripheral participation*. Cambridge: Cambridge University Press.
- Liu, M. (1998). A study of engaging high-school students as multimedia designers in a cognitive apprenticeship-style learning environment. *Computers in Human Behavior*, 14(3), 387–415.
- McCombs, B. L. (2001). Self-regulated learning and academic achievement: A phenomenological view. In B. J. Zimmerman & D. H. Schunk (Eds.), *Self-regulated learning and academic achievement: Theoretical perspectives* (pp. 67–123). Hillsdale: Lawrence Erlbaum Associates.
- Mislevy, R. J., & Gitomer, D. H. (1996). The role of probability-based inference in an intelligent tutoring system. *User-Modeling and User-Adapted Interaction*, 5, 253–282.
- Mislevy, R. J., & Riconscente, M. (2005). *Evidence-centered assessment design: Layers, structures, and terminology* (PADI Technical Report 9). Menlo Park: SRI International.
- Mislevy, R. J., Steinberg, L. S., Breyer, F. J., Almond, R. G., & Johnson, L. (1999). A cognitive task analysis with implications for designing a simulation-based assessment system. *Computers in Human Behavior*, 15, 335–374.
- Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2002). Design and analysis in task-based language assessment. *Language Testing*, 19, 477–496.
- Mislevy, R. J., Steinberg, L. S., & Almond, R. G. (2003). On the structure of educational assessments. *Measurement: Interdisciplinary Research and Perspectives*, 1, 3–67.
- Nelson, B., Erlandson, B., & Denham, A. (2010). Global channels for learning and assessment in complex game environments. *British Journal of Educational Technology*. Published online, January 2010. To appear in print, June 2010.
- Perry, N. E., VandeKamp, K. O., Mercer, L. K., & Nordby, C. J. (2002). Investigating teacher-student interactions that foster self-regulated learning. *Educational Psychologist*, 37(1), 5–15.
- Pintrich, P. R., Smith, D. A. F., Garcia, T., & McKeachie, W. J. (1993). Reliability and predictive validity of the motivated strategies for learning questionnaire (MSLQ). *Educational and Psychological Measurement*, 53, 801–813.
- Plass, J. L., O'Keefe, P., Homer, B. D., Case, J., Hayward, E. O., Stein, M., & Perlin, K. (in press). Motivational and educational outcomes associated with individual, competitive, and collaborative game play. *Journal of Educational Psychology*.
- Plass, J. L., Homer, B. D., Hayward, E. O., Frye, J., Huang, T. T., Biles, M., Stein, M., & Perlin, K. (2012, September 18–20). The effect of learning mechanics design on learning outcomes in a computer-based geometry game. *Paper presented at GameDays 2012* held, at Fraunhofer IGD TU Darmstadt, Darmstadt, Germany.

- Roth, E. M., & Woods, D. D. (1989). Cognitive task analysis: An approach to knowledge acquisition of intelligent system design. In G. Guida & C. Tasso (Eds.), *Topics in expert system design* (pp. 233–264). New York: Elsevier Science.
- Rupp, A. A., Gushta, M., Mislevy, R. J., & Shaffer, D. W. (2010). Evidence-centered design of epistemic games: Measurement principles for complex learning environments. *Journal of Technology, Learning, and Assessment*, 8(4). Retrieved June 1, 2011 from <http://www.jtla.org>
- Salen, K., & Zimmerman, E. (2003). *Rules of play: Game design fundamentals*. Cambridge: MIT Press.
- Schraagen, J. M., Chipman, S. F., & Shalin, V. L. (Eds.). (2000). *Cognitive task analysis*. Mahwah: Lawrence Erlbaum Associates.
- Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational Psychology Review*, 7(4), 351–371.
- Schwartz, D. L., & Black, J. B. (1990). *The induction of rules from analog, mental models*. Paper presented at the annual meeting of the American Educational Research Association, Boston, MA.
- Shaffer, D. W. (2006). *How computer games help children learn*. New York: Palgrave Macmillan.
- Shute, V. J. (2010). *Innovative assessment for the 21st century: Supporting educational needs*. New York: Springer.
- Shute, V. J. (2011). Stealth assessment in computer-based games to support learning. In S. Tobias & J. D. Fletcher (Eds.), *Computer games and instruction* (pp. 503–524). Charlotte: Information Age.
- Shute, V. J., & Kim, Y. J. (2011). Does playing the World of Goo facilitate learning? In D. Y. Dai (Ed.), *Design research on learning and thinking in educational settings: Enhancing intellectual growth and functioning* (pp. 243–267). New York: Routledge Books.
- Shute, V. J., & Torrealano, L. A. (2003). Formative evaluation of an automated knowledge elicitation and organization tool. In Murray, T., Ainsworth, S., & Blessing, S. (Eds.), *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive, and intelligent educational software* (pp. 149–180). The Netherlands: Kluwer Academic Publishers.
- Sicart, M. (2008). Defining game mechanics. *Game Studies*, 8(2). Retrieved June 1, 2011 from <http://gamestudies.org/0802/articles/sicart>
- Spiro, R. J., Coulson, R. L., Feltovich, P. J., & Anderson, D. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. In V. Patel (Ed.), *Proceedings of the 10th annual conference of the cognitive science society*. Hillsdale: Erlbaum.
- Spiro, R. J., & Jehng, J. C. (1990). Cognitive flexibility and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter. In Nix, D. & Spiro, R. (Eds.), *Cognition, education, and multimedia: Exploring ideas in high technology* (pp. 163–205). Hillsdale: Erlbaum.
- Swink, S. (2008). *Game feel: A game designer's guide to virtual sensation*. New York: Morgan Kaufmann.
- Tychsen, A., & Canossa, A. (2008). Defining personas in games using metrics. In *Proceedings of the 2008 conference on future play: Research, play, share* (pp. 73–80). New York/Boston: Association Computing Machinery.
- Um, E., Plass, J. L., Hayward, E. O., & Homer, B. D. (2011). Emotional design in multimedia learning. *Journal of Educational Psychology*, 104(2), 485–498.
- Williamson, D., Bauer, M., Steinberg, L. S., Mislevy, R. J., Behrens, J., & DeMark, S. (2004). Design rationale for a complex performance assessment. *International Journal of Testing*, 4, 303–332.
- Winne, P. H. (1982). Minimizing the black box problem to enhance the validity of theories about instructional effects. *Instructional Science*, 11, 13–28.

Chapter 32

Conceptually Meaningful Metrics: Inferring Optimal Challenge and Mindset from Gameplay

Carrie Heeter, Yu-Hao Lee, Ben Medler, and Brian Magerko

Take Away Points:

- This chapter describes our experience constructing *conceptually meaningful gameplay metrics* that are helpful in exploring *which players are likely to receive the intended benefits* from playing serious games.
- Takeaways and lessons learned from two case studies are presented in which the authors define, construct, and apply game metrics to measure conceptually meaningful player behaviors.
- Case study 1 looks at brain games played repeatedly over a period of weeks, addressing the question of how to construct gameplay metrics that offer insight into whether brain game players are experiencing the optimal level of challenge, and hence are likely to experience the cognitive benefits they were seeking.
- Cast study 2 considers gaming mindset and learning game play. Mindset theorist Carol Dweck has shown that people with a fixed mindset, who believe intelligence is fixed and can't improve with effort, avoid hard challenges because they interpret failure as a negative evaluation of their intelligence (Dweck 2000).

C. Heeter (✉)

Department of Telecommunication, Information Studies, and Media,
Michigan State University, 2467 Funston Avenue, San Francisco, CA, USA
e-mail: carrie.heeter@gmail.com

Y.-H. Lee

Media & Information Studies (PhD Program), Michigan State University,
404 Wilson Road, Room 409, East Lansing, MI, USA
e-mail: leeyuhao33@gmail.com

B. Medler

Chief Creative Office, Electronic Arts, Redwood City, CA, USA

B. Magerko

Digital Media Program, Georgia Tech, Atlanta, GA, USA

Many significant gameplay differences were found between fixed mindset and growth mindset players, with likely implications for whether and how much each learned from the game.

32.1 Introduction

In this chapter we describe our experience with constructing conceptually meaningful gameplay metrics that are helpful in exploring which players are likely to receive the intended benefits from playing serious games. We discuss takeaways and lessons learned from two case studies in which the authors sought to define, construct, and apply conceptually meaningful game metrics.

Unlike lectures or textbooks where all learners are exposed to the same learning stimulus, in games the actions and choices made by each player create a unique experience each time a game is played. Many games allow players to choose difficulty levels, players may play for different amounts of time, and games are often designed to become more challenging as the game progresses. Player skill and choices magnify the differences in what each individual player experiences.

The metrics discussed in this chapter were grounded in theory, including flow and optimal challenge in brain games (Csíkszentmihályi 1990) and gaming mindset in a learning game (Dweck 2006). For each study we explain the underlying theory, discuss how the metrics were developed and we found by examining play behavior. We discuss how these metrics can help to identify play patterns of vulnerable players (such as those with little gaming experience or those who enjoy exploration but not competition, yet are assigned to learn from a competitive game) who may be disadvantaged in the context of being assigned to play a serious game. We present how such metrics helped us envision improved gameplay options for those players.

For each case study, we begin with a brief introduction to the measurement challenge. For Case Study 1, the key issue is how to design metrics to measure flow and detect when a player is playing at his or her optimal level of challenge. For Case Study 2, the key issue is how to design metrics to detect player mindset in a learning game, with the eventual goal to adapt the game to better suit player mindset. Key takeaways are summarized. Next, the theoretical background is explained in more detail. We then describe how we went about designing game metrics related to the theory. The final section for each case study explores example user data from actual players and considers whether the metrics were useful and informative. Each case study concludes with implications for metric and game design.

32.2 Case Study 1: Measuring Player Experiences of Flow and Optimal Challenge in Brain Games

Brain games are designed to exercise cognitive functions such as working memory, language, and visual-spatial skills, with the goal of increasing cognitive reserves (the brain's resilience and ability to recover from damage (Fernandez 2007a)) and

improving cognitive functions. It turns out that simply playing brain games, even playing a lot, doesn't necessarily result in the intended cognitive gains. Unlike cognitive exercises prescribed by a neurologist to precisely match the abilities and needs of an individual patient, brain games are typically designed to allow players to select which games they play, how often, and at what difficulty level. Although it may be enjoyable, facing brain games the player is already proficient at and choosing easy levels probably will not result in cognitive gains.

Frequency and duration of play is not sufficient for cognitive gain, if the player only chooses easy challenges and succeeds easily.

Cognitive scientists have shown in controlled experiments that exercising the brain by doing tasks that are just challenging enough – not too hard and not too easy – over a period of days or weeks can increase brain plasticity and stimulate the brain to grow new neural pathways and improve cognitive skills (Hardy and Scanlon 2009). In this section we explore brain game metrics that can offer insight into whether brain game players are experiencing the optimal level of challenge when they play.

Key takeaways of this section are:

- To address the potential for cognitive gain, it is necessary to look at brain game play over time rather than examining individual game play or average play behaviors.
- We decided upon a strategy to compare average play in the first and last week of a study where subjects played for up to 6 weeks.
- Accuracy, speed, and self-selected difficulty level are the core building blocks for measuring brain game play.
- Frequency and duration of play is not sufficient for cognitive gain, if the player only chooses easy challenges and succeeds easily.
- In general, we expect to see improvements in accuracy and speed over time, if the player is experiencing cognitive gains.
- In general, we expect to see players select more difficult challenges over time, in order to experience cognitive gains.
- Selectable difficulty level complicates comparisons of accuracy and speed. More difficult levels result in lower accuracy and longer time to solve the game.
- High accuracy success (90% or higher) leaves little room for sufficient challenge to evoke cognitive gain.
- In retrospect, good-for-the-brain brain games may need to be designed with more failure and harder (usefully harder, exercising the targeted cognitive function) challenges than games for entertainment.

32.2.1 Theoretical Background

Age-related cognitive decline had long been considered inevitable and irreversible, including declines in processing speed, sensory integration, and memory function. New research has reversed conventional wisdom, uncovering surprising potential

for brains of all ages to change (e.g., Mahncke et al. 2006). Research now shows that learning (and living) constantly changes the brain, a concept known as brain plasticity, which refers to the lifelong capacity for physical and functional brain changes. Brain plasticity can have a positive effect, restoring to some degree the normal declines seen in cognition due to aging or neurological impairment, or may have a negative effect (degrading brain function) over time. Research on brain plasticity implies that benefits from playing cognitive games derive from exercising diverse cognitive domains (Fernandez 2007b) and from taking on challenges just difficult enough to stretch existing abilities (Hardy and Scanlon 2009; Fernandez 2007b). In other words, which games a player chooses and the way a player plays a brain game is expected to influence whether and how much playing that game contributes to improved cognitive function.

Willis et al. (2006) conducted and published the first large-scale, randomized trial showing that cognitive training improved cognitive function in older adults and that the improvement lasted up to 5 years after the intervention. Their Advanced Cognitive Training for Independent and Vital Elderly (ACTIVE) project exposed study participants to one of three 10-session, 60–75 min training interventions each narrowly targeting a specific cognitive ability (short term memory, reasoning, or visual search and divided attention). Mahncke et al. (2006) also exposed older adults to memory training and documented benefits of brain plasticity-based interventions targeting normal age-related cognitive decline.

Compared to the promising results of cognitive training, clinical studies examining computer-based brain games offer mixed results. Owen and colleagues (2010) assigned young and middle aged individuals to play 4 h of brain games during a 6-week period (24, 10-min sessions). They found no compelling evidence of effects. On the other hand, Scanlon et al. (2006, 2007a, b) found evidence of improvement in working memory, visual attention, and executive function among study participants who played *Lumosity*¹ brain games (Lumos Labs, Inc., 2012) 20 min per day for 5 weeks. Additionally, our own results suggest possible selective improvement of episodic memory from playing memory games (Heeter et al. 2008; Bozoki et al. Under review).

Cognitive exercises are a mental equivalent of physical therapy. A neurologist may recommend that a patient follow a specific cognitive exercise routine, specifying frequency and duration of exercise as well as exercises to complete. Unfortunately, low patient compliance with physical and cognitive therapy exercise prescriptions is a common problem (Sluijs et al. 1993). Brain games attempt to counteract this limitation. With their dual promise of fun and cognitive exercise, brain games are designed to entice players to play in ways that benefit their brain. Therefore, in a way, brain games more closely resemble a gym or health club than physical therapy, because members choose how often to show up and what activities to engage in; whereas, physical therapy sessions are carefully prescribed. Once patients are enticed by the less rigid nature of brain games, a critical factor is to try to ensure that

¹ <http://www.lumosity.com/>

participants challenge themselves sufficiently during voluntary activity to gain the desired physical or cognitive benefits.

Brain games as leisure activities need to be pleasant enough to play to attract and retain players, but without sacrificing its central goal of cognitive exercise. New neural connections are more likely to arise when a player encounters just the right level of challenge (Hardy and Scanlon 2009). Therefore, brain gameplay for each individual should ideally be hard enough to evoke what Gee (2007) describes as “pleasant frustration”, but not so hard as to be impossible for that player to succeed.

Serious games intended to positively affect brain plasticity should monitor a player’s current skill level and might adapt the challenge level presented in the game. One theory related to the skill level of a given person in regards to a certain activity is the theory of flow. Through his research on happiness, Csíkszentmihályi (1990) defined the psychological state of “flow” as an ideal, euphoric state evoked when challenge and personal ability are in optimal balance. If the challenge is too little, the result is boredom. If the challenge is too great, the result is anxiety. Flow state occurs when eight conditions are met: achievable goals, concentration on the task, clear goals and timely feedback, deep but effortless involvement, a sense of control, reduced sense of self, and altered sense of time (Salen and Zimmerman 2004, p. 337, discussing flow in the context of game design). Using these conditions as guidelines, along with other research on using flow in games, our study attempts to create a set of game metrics to monitor a player’s current skill level and whether or not she is experiencing challenge hoping to predict whether that player is likely to be receiving positive cognitive benefits.

Flow and optimal challenge for cognitive growth sound strikingly similar. But are they the same construct? Aldrich (2010) proposes that one way to measure flow is expertise – how quickly a player can complete a level. An expert can finish a level more quickly than a novice. Beume and colleagues (2008) also equated flow with response speed. They operationalized flow in Pac-Man as “the interaction time fraction between the human-controlled Pac-Man and the ghosts.” More skillful players perform better, and therefore were assumed to be experiencing flow.

Challenge, in brain games, is partially under the control of the player. Like many entertainment games, brain games typically offer players a choice of difficulty level. Games from the commercial brain game company, *Happy Neuron*² offers between three and six difficulty levels (Happy Neuron 2012a). Within a chosen difficulty level, some brain games are designed to get progressively harder as the player succeeds. Progressive difficulty in a game is controlled by the game, not by player choice. When gameplay is too hard, players can reduce the challenge they are facing by exiting the game and starting over at an easier level or by playing a different game. Optimal self-challenge is unique to each individual. Optimal self-challenge is also dynamic, dependent upon players’ current ability as well as their current physical and mental state. Optimal self-challenge is voluntary in that players select

² <http://www.happy-neuron.com/>

Fig. 32.1 Headline Clues



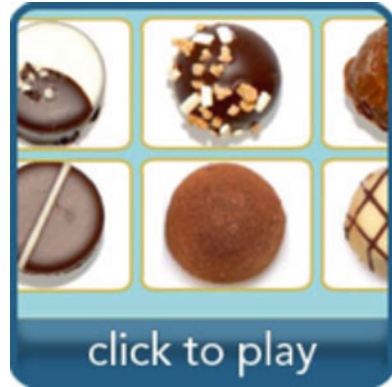
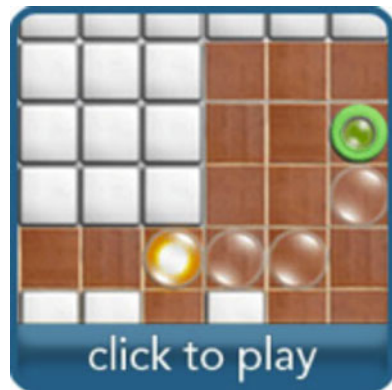
a difficulty level and decide to keep playing. To experience optimal self-challenge, players must choose to play at a level slightly beyond what they can easily win.

Skillful, nearly effortless success is not the recipe for cognitive improvements from playing brain games. Quite the contrary, brain gain is best if the player has to make efforts and struggle to succeed. Optimal self-challenge for cognitive gain in brain games probably feels a bit more frustrating and less pleasant than flow. Unlike games for entertainment, the point of brain games is cognitive gain. Pleasure and fun while playing brain games help players want to keep playing, but they are secondary to optimal self-challenge if the goal is to achieve cognitive benefits.

32.2.2 Game Metrics

This section discusses the metrics we designed for creating and studying three games that were part of Brain Powered Games, a suite of cognitive games intended to exercise one or more different brain functions. Brain Powered Games are games developed at Michigan State University: Games for Entertainment and Learning (GEL) Lab. *Headline Clues* (Michigan State University: Games for Entertainment and Learning (GEL) Lab, 2008), shown in Fig. 32.1, was designed to exercise verbal and language skills, similar to a crossword puzzle but focused on current events. Players solve headlines with missing words and letters about the news of the day. *Keep It In Mind* (Michigan State University: Games for Entertainment and Learning (GEL) Lab, 2008), shown in Fig. 32.2, exercises working memory and mental focus. *Sokoban* (Michigan State University: Games for Entertainment and Learning (GEL) Lab, 2008), shown in Fig. 32.3, is a logic puzzle game that exercises strategic planning and uses both reasoning and visual-spatial skills. The objective of the game is to push lenses through a maze and onto targets in the minimum number of moves possible.

Accuracy, speed, and self-selected difficulty level are the core building blocks for measuring brain game play. As background steps for measuring optimal challenge,

Fig. 32.2 Keep It In Mind**Fig. 32.3** Sokoban

we will discuss specific examples of how speed, accuracy, and other data were operationalized for the three brain games introduced above.

Web services offering online brain games need a consistent way of keeping score and giving subscribers feedback about their performance over time across all of the current and potential future brain games, even though the games are diverse in terms of cognitive domain, game mechanics, and game goals. For example, a “guess the missing words” headline game is very different from a memory game, which is very different from a Sudoku game. Lumosity lets players track their “BPI” (Brain Performance Index) over time (Lumosity 2011). Each time a subscriber plays a game her BPI is updated with a new entry for that day. *Happy Neuron* uses speed and accuracy to calculate and compare scores, weighted by age and gender. Scores ranging from 0 to 100 are assigned based on speed and accuracy relative to the performance of an average person with a similar profile. If you are the same as an average person, your score is 50. Performance is calculated based on standard deviations, assuming a normal curve (Happy Neuron 2012b). Happy Neuron and Lumosity continuously collect player data and use that data to update their comparison data with ever more accurate averages.

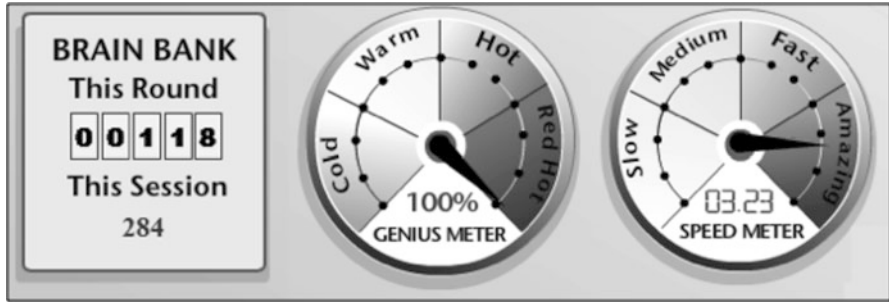


Fig. 32.4 Core scoring interface for brain powered games

For these Brain Powered Games, player accuracy was calculated as the percent success they achieved while playing one round or solving one puzzle. Players saw their performance on a ‘Genius Meter’ at the end of each round. The genius meter sits beside a speed meter, shown in Fig. 32.4. Speed is rated slow, medium, fast, or amazing based on typical expectations of how quickly a moderately fast player can complete the game. The speed meter value ranges from 0 to 100, just like genius. Algorithms unique to each game calculate “Brain Bank” points based on the player-selected difficulty level, their genius score and their speed score. (In a few instances, other factors may contribute to total brain bank points.)

Deciding how to compute genius percent required us to integrate game performance variables into a single metric. Doing so was easy for the memory game because the task in each round was to remember some number of items. The game knew exactly how many items the player was supposed to remember and how many they got right. So calculating genius percent was straightforward (number of wrong answers divided by total items). For the headline word game, solving the headline perfectly without requesting any hints or making any mistakes was clearly worth 100% genius. But how should wrong guesses and asking for hints be combined in a formula to calculate genius percent? How much worse (ungenius-like) is it to guess wrong than to ask for a hint?

Selectable difficulty level complicates comparisons of accuracy and speed. The user-selected difficulty level controlled the number (and length) of missing words in a headline. Easy headlines had two missing words, medium headlines had three missing words, and hard headlines had four missing words. Because of the difficulty level, we knew the number of missing words and the number of missing letters. Also, you could solve a headline by requesting a hint for every missing letter, so we knew that winning by requesting a hint for every missing letter should be worth zero genius percent. The final formula to calculate the genius percent in *Headline Clues* was based on the following formula (any value less than zero is set to zero):

$$\text{Genius}\% = 1 - \left(\frac{\left(\frac{\text{HintsRequested}}{\text{MissingLetters}} \right) + \left(\frac{\text{WrongGuesse}}{\left(\text{MissingWords} * 2 \right)} \right)}{\right)$$

Table 32.1 Brain powered games variable codebook

Level	Round number played in a multiple round game such as 1–5
Speedp	Speed percentage (0–100)
Geniusp	Genius percentage (0–100)
Brainp	Brain bank points earned
Difficulty	Player- selected level of difficulty, where 1 = gentle, 2 = medium, 3 = power
Duration	Total duration of round in milliseconds
Solved	Was this round won or solved where 1 = true and 0 = false
Special1	Sokoban = total moves used; HClues = clues Used;
Special2	KIIM = memory goal for the game (player sets in the beginning) Sokoban = minimum moves; HClues = wrong guesses;
Special3	KIIM = number of items remembered in this round Sokoban = number of Restarts; HClues = number of missing letter words; KIIM = for power only, what the assignment was (remember in reverse alphabetical order, numerical order, order shown, etc.)

We used similar methods to formulate the genius percent for the other games. Designers determined what they expected successful players to do. We then found ways to quantify different aspects of expected performance into a 100% score.

Rating speed as slow, medium, fast, or amazing for easy, medium, and hard difficulties required many trials to estimate average solving speed. We needed to choose a value for each game and level that would be considered slow, a speed that would be considered amazing, and a formula to position players on the speed meter. Multipliers were chosen to balance the brain bank points to try to reflect user-selected difficulty and the game design difficulty so that brain bank points held their value and made sense across the games.

XML files for each game stored speed parameters and base multipliers for the different levels so that they could be easily adjusted until balance was achieved. Here is an example:

```
<timeGentle>3600</timeGentle>
<timeModerate>4320</timeModerate>
<timePower>5040</timePower>
<baseGentle>250</baseGentle>
<baseModerate>375</baseModerate>
<basePower>500</basePower>
```

Brain Powered Games stored data for each round of play in each game for ten shared variable names; some identical across games and others with slightly different meanings (see Table 32.1).

This measurement example shows how brain game designers make assumptions about what constitutes perfect play and how much value to place on different performance-related player actions. Initially, lots of trial and error is necessary to balance the games, adjusting and validating the fairness of these scoring decisions.

Brain Powered Games was a prototype service, so our assumptions in scoring were based on trials we conducted ourselves. Commercial services such as *Happy Neuron* and *Lumosity* are able to take advantage of ongoing subscribers' play sessions to collect data that can be used to continuously improve scoring.

32.2.3 *Examining Optimal Self-Challenge Over Time*

We worked with Michigan State University Neurology Assistant Professor Andrea Bozoki to conduct a longitudinal study of older adults who were recruited to play our games at least 5 days a week, half an hour per day, over 6 weeks. These older adults volunteered to participate knowing that the study involved some kind of intervention to exercise cognitive functions, but not knowing it involved games. All were cognitively healthy mature adults who were playing specifically with the goal of maintaining or improving brain function.

To address the potential for cognitive gain, it is necessary to look at brain game play over time rather than examining individual game play or average play behaviors.

With our game metrics system set to monitor players, the brain games were randomly assigned to all 40 participants (all 60–80 years old). Forty others were assigned to a control experiment. They read online news, listened to podcasts, or watched online videos for the same amount of time as the experimental group played games. Participants were asked to play any combination of five available games for at least half an hour per day, 5 days per week, for 6 weeks. All participants received incentive payments of up to \$60; \$5 for each week that they logged in to the program at least five times for at least 30 min, and an additional \$30 at the end of the study if total logged time exceeded 1,600 min. Despite the instructions and rewards, we did not end up with 6 weeks of gameplay data for 40 participants. A few only played for 1 week. Thirty participants played one or more days in at least each of 3 weeks. Thus, we decided as a strategy to compare average play in the first and last week of a study where subjects played for up to 6 weeks.

The game metrics system was set up to output an excel spreadsheet with one row of data for every round of every game every player played. The spreadsheet was 58,895 rows long, including game play data for all enrolled participants who played at least one round of one game. In order to analyze progression over time, we worked with psychology and neurology students to hand code each participant's data for each game. The day of each player's first instance of play was flagged. That day plus as many of the next six consecutive calendar days that a participant played the game were included in their first week averages for that game. The coder then moved to the last observed date of playing that game for that person. The final play day and the previous six consecutive calendar days' play were used to compute that person's

Table 32.2 Comparing first and last week averages

Week	Keep It In Mind			Headline Clues		Sokoban	
	Solved (%)	Difficulty	Items	Solved (%)	Difficulty	Solved (%)	Difficulty
First	71	1.5	3.4	90	1.7	86	1.2
Last	76	2.3	3.7	98	1.7	72	1.8

final week averages for that game. All data for that game between the first week and final week were included in that person's middle week's averages, regardless of whether that time frame was 1, 2, 3, or 4 weeks. This was calculated for all players who played *Headline Clues*, *Keep It In Mind*, and *Sokoban*.

In general, we expected to see improvements in accuracy and speed over time, if the player is experiencing cognitive gains. Additionally, we expected to see players select more difficult challenges over time, in order to experience cognitive gains. Table 32.2 shows change from the first week to the last week average percent of rounds solved or won, average player-selected difficulty (where 1=easy and 3=hard), and an additional difficulty measure for *Keep It In Mind*, level (average number of items remembered). Significant differences based on paired t-tests are noted in the paragraphs following the table (Table 32.2).

Headline Clues showed no significant differences between the first and last weeks of play. Players, on average, selected a difficulty of 1.7 (most chose either easy or medium), and they did not move up to harder challenges over time. The percent of headlines solved was high in the first week (90%) and almost as high as one can get (98% out of 100%) by the last week. In other words, those who played *Headline Clues* were good at it, but felt little need to increase the level of challenge to hard. Because of this, they probably received little if any improvements in verbal cognition due to gameplay. As it turns out, the hard level of the word game was too hard, so players almost never selected it!

Players who played *Keep It In Mind* chose significantly higher difficulty levels in the final week (2.3) than they had in the first week (1.5), $t(29)=-5.58$, $p<.001$. They remembered more items (3.4 in the first week and 3.7 in the final week, $t(29)=-2.80$, $p=.009$). Further, they solved significantly more of the challenges they attempted (76% compared to 71%) ($t(29)=-2.41$, $p=.022$). This pattern of play is exactly how experts hope brain games will be played, i.e., participants would seek harder challenges over time, attempted to remember more items, and, even with these harder challenges, by the final week they succeeded more often. It is reasonable to expect that cognitive gain in episodic memory occurred for players who fit the pattern.

Sokoban play also increased significantly in difficulty from the first to the last week, from an average of 1.2–1.8, $t(29)=-5.51$, $p<.001$. This means that players sought harder challenges over time. Their success in solving *Sokoban* puzzles was significantly different, but in the opposite direction than *Keep It In Mind*, $t(29)=-3.47$, $p=.002$. *Sokoban* players succeeded less often in the final week than in the first week. This likely is because the puzzles got much harder. The difficulty of *Keep It In Mind* never exceeded trying to remember seven items. *Sokoban* puzzles just kept

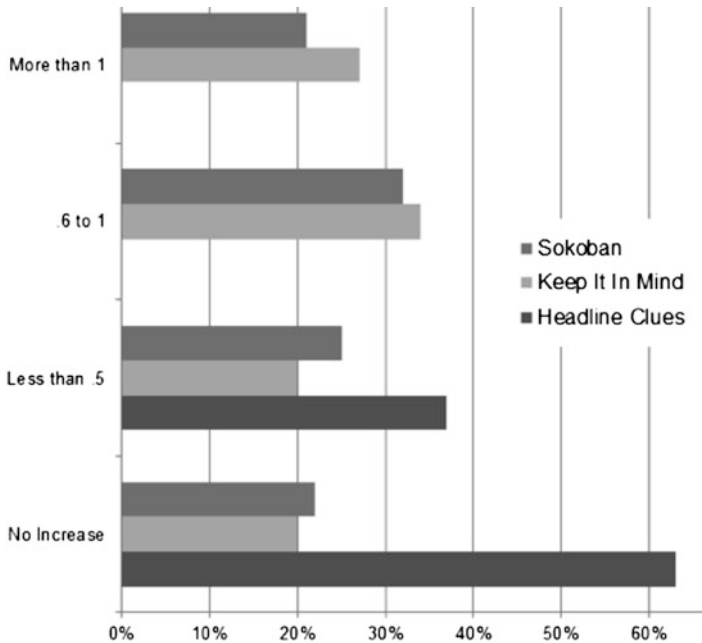


Fig. 32.5 Change in self-selected difficulty from first to last week

getting harder. The evidence that *Sokoban* players sought progressive challenge suggests that these players also received cognitive benefits from play.

The analysis in Table 32.2 compares player averages. To explore what actually happened for individuals, we looked at which players increased their self-challenge and by how much. Figure 32.5 shows the change in self-selected difficulty from the first to the last week. For *Keep It In Mind*, 20% of players did not increase difficulty level (in fact, 13 decreased their challenge from the first to the last week). Another 20% increased their selected difficulty level slightly during the study period (showing a change of less than 0.5 on a difficulty scale from 1 to 3). Those players (around 40%) probably did not experience memory improvements from gameplay. (The hypothesis that the best level of difficulty for achieving cognitive benefits is difficulty hard enough to be challenging, but not so hard that failure usually occurs needs to be examined through future research. In this chapter merely we show how game metrics can be used to classify experienced difficulty.) 34% increased difficulty by about one whole notch, and 27% greatly increased their self-challenge. A similar pattern was found for *Sokoban*. But in the third game, *Headline Clues*, players mostly did not change their difficulty level from week 1 to the last week.

Next we looked at the relationship between performance and difficulty in the first week of play and in the last week of play. In general, performance was better at easier difficulty and worse at harder difficulty, at least for some players. Here is the basic formula we used for looking at self-challenge over time in brain games. We grouped players into four levels of time spent playing the game in the final

week. Also, we grouped players into four levels of change in self-selected difficulty from the first to the last week of play. The result is a 4×4 grid (time spent by increase in challenge). We compared average percent of puzzles attempted that were successfully solved in the first week and in the last week, for each of the 16 cells in the grid, looking at whether there is evidence for self-challenge likelihood that cognitive improvements occurred due to gameplay.

The *Sokoban* analysis was the most straightforward because there was a single measure of performance: whether or not the player succeeded in solving the puzzle. Table 32.3 shows the percent of puzzles attempted that were solved in week 1, the percent of puzzles solved in the final week, and the change in percent of puzzles solved from week 1 to the final week. The expectation is that players who take on harder challenges will receive more cognitive benefits than those who take on easy challenges.

One indication of challenge is self-selected difficulty level. Another view of challenge is how often the player succeeds. Increased self-selected challenge and lower percent of puzzles successfully solved are both indications that the player is taking on difficult challenges. Playing for a small amount of time is another indication the player is exerting little effort.

Games which provide chances to fail more often, and player tolerance of failure in pursuit of cognitive gain, are likely to be key elements to effective design and use of brain games.

In retrospect, good-for-the-brain brain games may need to be designed with more failure and harder (usefully harder, exercising the targeted cognitive function) challenges than games for entertainment.

Applying that logic to the cells in Table 32.3, we have shaded the quadrant least likely to experience brain gain in gray (those who played for the least time, at the easiest difficulty level) and the quadrant most likely to experience gain in white (those who played for longest and increased difficulty the most). Curiously, those, who took on increasing challenges, experienced a dramatic decrease in the percent of puzzles solved correctly. They apparently were willing to tolerate repeated failures in pursuit of challenge. The group who played the longest and increased their challenge the most had a shockingly low, 17% success rate. Their success rate dropped by 73% from the first to the last week. And yet, this is probably exactly the kind of play neurologists would want to encourage. On the opposite cell in the table, those who played for the least amount of time, at the lowest challenge, experienced slight gains in success rate. They are unlikely to have received any cognitive benefit from playing. If we had more data, ideally tied to cognitive testing, we could recommend an optimal failure rate *Sokoban* should aim for to maximize brain gain.

Self-challenge in *Keep It In Mind* is more complicated to measure than self-challenge in *Sokoban*, because *Keep It In Mind* players had several ways they could control their challenge. Table 32.4 presents two of them: average percent of rounds

Table 32.3 Percent of Sokoban puzzles solved and change in average selected difficult

Final week playing time	Increase in difficulty, week 1 to final week				n
	None	Slight	Moderate	Large	
<16 min					6
Week 1	67%	78%			
Final week	69%	89%			
% change	2%	11%			
16–52 min					7
Week 1		93%	86%	93%	
Final week		78%	74%	69%	
% change		–15%	–12%	–24%	
52–68 min					7
Week 1	89%	95%	96%		
Final week	70%	89%	88%		
% change	–19%	–6%	–8%		
>68 min					6
Week 1		81%	94%	90%	
Final week		84%	66%	17%	
% change		3%	–28%	–73%	
Avg. difficulty (final week)	1.0	1.7	2.1	2.5	
n	6	7	7	6	26

correctly solved (i.e., the player successfully remembered however many items they were asked to recall) and the average number of items they tried to solve. Players could set a goal (between 3 and 7 items). They then had to correctly solve 2, then 3, then 4, and so on, up to their number of items they had selected as their memory goal. When they failed to remember all of the items in a round, players could repeat the current number of items, go back to one fewer item, or start a new game. Thus, the average items remembered is an amalgam of player choices and the game design. One indication of challenge is self-selected difficulty level. Another view of challenge is how often the player succeeds. A third view of challenge is how many items the player tried to remember. Increased self-selected challenge, lower percent solved, and higher number of items remembered should be associated with more cognitive gain.

Applying that logic to the cells in Table 32.4, we have again marked quadrants likely not to experience brain gain in grey and those who may experience gain in white. Playing for fewer than 10 min seemed insufficient to have an impact on memory. Playing for at least 19 min in the final week and moderately or greatly increasing their self-selected challenge between week 1 and the last week would be expected to be associated with cognitive gain, if our hypothesis about optimal self-challenge is correct. It is interesting to note that that group had fairly low success rates, ranging from 65 to 79% solved. Our reason for supposing they probably improved was their commitment to play that was difficult enough (for them) that they failed quite often.

After examining player behavior over time, we discovered that people who played the *Headline Clues* game over the study period never increased their chal-

Table 32.4 Percent of Keep It In Mind puzzles solved and average change in selected difficulty

Final week playing time	Increase in difficulty, week 1 to final week				n
	None	Slight	Moderate	Large	
<10 min					6
Week 1	83%	76%	89%		
Final week	90%	75%	79%		
Change	7%	-1%	-10%		
10–18 min					8
Week 1	61%	54%	69%		
Final week	73%	63%	71%		
Change	12%	9%	2%		
19–30 min					7
Week 1	78%		66%	65%	
Final week	87%		79%	67%	
Change	9%		13%	2%	
>30 min					8
Week 1		64%	69%	73%	
Final week		57%	77%	79%	
Change		-7%	8%	6%	
Average difficulty (final week)	1.6	2.2	2.5	2.8	
n	7	7	8	7	29

allenge level. Therefore we could not use the approach just described for Soboban and Keep It in Mind to identify players who seemed to be playing at an optimal level of self-challenge to improve their language cognitive function. We did not divide players into four groups of increased challenge from the first to the final week, because players did not change their challenge levels. Table 32.5 shows the percent of headline puzzles correctly solved in the first and final weeks by duration of play in the final week. Those, whose play behaviors might be expected to result in brain gains, should show a large jump in percent of puzzles solved correctly. The only players whose solving percentage increased notably between the first and final week were those who played *Headline Clues* for fewer than 25 min in the final week. Their initial solving rate (80%) increased significantly, averaging 95% in the final week ($t[26]=2.945$, $p=.007$). Those who played the longest (nearly half of the participants) started out nearly perfect in the first week (95–97% of puzzles solved) and ended up with 99% of puzzles solved. In other words, there was almost no room for improvement.

High accuracy success (90% or higher) leaves little room for sufficient challenge to evoke cognitive gain.

In Sect. 32.2.1 (Theoretical Background), we chose self-challenge over the idea of flow (defined by Aldrich (2010) and Beume et al. (2008) as the speed with which one completes a level) as the best recipe for brain gain. From a metrics standpoint, it would not have been meaningful to compare level completion speed (flow) across players in our first two games because speed would be influenced by difficulty level as well as player skill. Some players changed their

Table 32.5 Headline Clues puzzles by final week minutes played

Final week playing time	Week 1 to final week changes			n
	% solved	Speed	Last week difficulty	
<25 min			1.6	6
Week 1	80	109		
Final week	95	72		
Change	15	-37		
25–38 min			1.8	8
Week 1	89	122		
Final week	100	85		
Change	11	-37		
39–76 min			1.7	7
Week 1	95	112		
Final week	99	62		
Change	4	-50		
>76 min			1.8	6
Week 1	97	114		
Final week	99	70		
Change	2	-44		

self-selected difficulty levels, choosing harder challenges in later weeks for the previous two games. However, *Headline Clues* players did not increase their self-selected difficulty levels between the first and final week. Therefore the change in solving speed could be compared. Table 32.5 shows that players solving speed became significantly and substantially faster between the first and final week ($t[25] = 7.872$, $p < .001$). They played between 37 and 50 s faster in the final week. Although most players' accuracy had little room to improve, players did solve the puzzles more quickly after weeks of playing. Improvements in solving speed demonstrate that players have gotten better at playing the game. Improvements in solving speed, controlling for challenge level, might be a better metric for predicting cognitive gain than looking at failure rate. Future research should examine the relationship between improvements in solving speed and improvements in verbal cognition.

32.2.4 Metric and Design Implications

In this section we summarize the case study presented and present the challenge of how to characterize optimal self-challenge in brain game players who played repeatedly for up to 6 weeks. We developed a grid to characterize player self-challenge over time based on effort (minutes player per week), average change in self-selected challenge level between the first and final week, and average performance across

rounds played in the final week. We applied this schema and identified quadrants of players in the grid who appeared to have engaged in the kind of self challenge neurologists expect would result in cognitive benefits for two of the games (Keep It In Mind and Sokoban). The third game had no change in self-challenge over time. All four levels of effort (minutes played) succeeded in solving Headline Clues puzzles substantially faster in the final week than they did in the first week. They clearly improved at playing the game. However, the gameplay data alone cannot confirm whether this improvement corresponded to growth of verbal cognition, or whether players simply had refined their strategies for how to beat the game.

We are currently working to compare cognitive performance testing results with our predictions based on gameplay data, to see whether the players we expect to have improved did so. We began by rejecting the proposition that flow state (measured as rapid, engaged, successful play) was the best play state for cognitive growth and instead proposed that self-challenge, and less than perfect initial performance were more likely to be associated with cognitive gain. Future research looking at cognitive gain in relation to self-challenge should address this question.

32.3 Study 2: Measuring Mindset in a Learning Game

Mindset theorist Carol Dweck has shown that people with a fixed mindset (about 40% of the population), who believe intelligence is fixed and can't improve with effort, avoid hard challenges because they interpret failure as a negative evaluation of their intelligence (Dweck 2000, 2006). In this section we examine gameplay of fixed-mindset and growth-mindset players, looking for expected differences that might interfere with learning for fixed-mindset players.

Key takeaways of this section are:

- We define and collect gameplay data from games we did not design ourselves.
- Our variables include frequency, duration, achievements, and player choices as totals, averages, or percentages.
- Duration of play complicates metric design. Enforcing duration of play is conducive to comparable metrics.
- Gaming mindset is measured using just four survey questions.
- Fixed mindset players won significantly fewer cases overall and they won a smaller proportion of the cases they brought to trial.
- Fixed mindset players made incorrect decisions for significantly more of the potential clients they encountered.
- Players with a 'growth' gaming mindset spent more time on performance and learning feedback.
- Players with a growth gaming mindset were significantly more likely to purchase upgrades that made gameplay more challenging.

32.3.1 *Theoretical Background*

People's intelligence may affect one's learning outcomes. However, studies of the implicit theory of intelligence shows that people's personal beliefs about the malleability of their intelligence will also affect how they approach learning, and consequently, how well they learn (e.g., Dweck 2000, 2006). The implicit theory of intelligence proposes two general types of learning mindsets—growth mindset and fixed mindset. Individuals with growth mindset (“incremental theorists”) believes that intelligence can be developed through devoting more efforts, while individuals with fixed mindset (“entity theorists”) believes that intelligence is fixed or developed at early age, intelligence cannot be improved once it is developed.

These two general mindsets affect how individuals bounce back and learn from their mistakes and how likely they are to seek new challenges. For individuals with a growth mindset, mistakes are perceived as opportunities to learn and improve. But for fixed-minded individuals, mistakes indicate a lack of ability (Dweck 2006).

Learning through digital games often requires learning through repeated play, trial and error (Gee 2007). Individuals with growth mindsets are more likely to benefit from this form of learning because they are not afraid of failure and pay close attention to learning from their failures so that they can improve. In contrast, individuals with fixed mindsets are likely disadvantaged learners when it comes to digital game based learning. Fixed mindset individuals are more likely to quit after experiencing failure. They quit because they believe failure is a result of their inability, and there is nothing they can do to improve it. Even worse, fixed mindset individuals are also more likely to cheat so that they do not experience failure at all (Blackwell et al. 2007). They tend to stick with familiar challenges to avoid failure.

Dweck (2006) posits that people may have different mindsets in different domains of experience. For example, one can have a fixed mindset related to performance in school, and growth mindset related to playing basketball. We adapted the four-item scale Dweck uses to measure mindset, asking specifically about gaming intelligence rather than intelligence overall (Lee et al. 2012). Our research confirms that differences in game-related mindset do affect learning behaviors and outcomes (Lee et al. 2012). When people with a fixed gaming mindset fail, they are more likely to lose attention overall, and they pay less attention to potentially corrective feedback. Fixed gaming mindset individuals are also less likely to actively seek challenge. As a result, individuals with a fixed gaming mindset often do not learning from mistakes and perform worse than growth gaming mindset individuals (Lee et al. 2012).

Studies in neural psychology have shown that people with the two mindset orientations respond differently to learning. Mangels et al. (2006) used brain imaging to compare how people with fixed or growth mindsets reacted to feedback about failure. The participants answered a large set of trivia questions and were given two kinds of feedbacks – performance feedback (how well they did) and learning feedback (the correct answers). Both groups paid attention to performance feedback, but those with a fixed mindset were fixated on the performance feedback and did not

Fig. 32.6 Do I Have a Right:
by iCivics and FILAMENT
Games. (<http://icivics.org>)



attend to the learning feedback. As a result, on a surprise re-test, growth mindset participants improved but fixed mindset participants made the same errors again. In other words, a fixed mindset can be an obstacle to learning from mistakes.

32.3.2 *Game Metrics*

The mindset metrics study focused on *Do I Have a Right* (Filament Games 2010), a learning game about the United States constitution. The game was designed by law and educational scholars from Arizona State University and developed by Filament Games. Players play as the owner of a new law firm and their job is to match clients with various legal problems to lawyers specializing in different amendments. When players correctly match a case, they earn prestige points, which can be used to hire more lawyers or buy upgrades that can either make the game more challenging or easier. A failed match results in losing cases. Learning takes place by reading about the amendments descriptions during gameplay and reading feedbacks about correct or incorrect matches between rounds. A complete game session constitutes of seven game days (rounds), but players can repeat the game many times to seek better performance (Fig. 32.6).

This game was chosen because:

- it is an effective learning game (Filament Games 2010);
- Players are exposed to the core game mechanics and considerable learning content early on;
- it was built in Flash; and
- the company granted us access to the source code to collect gameplay data for constructing metrics.

32.3.2.1 **Gameplay Data Collection System**

We defined and collected gameplay data from games we did not design ourselves

We created a proprietary system to collect gameplay data from the third party Flash games. We embed a metric application programming interface (API) within a game's Flash source code to capture game events, sending gameplay event data to a secure MySQL database. Like many of the telemetry systems discussed in this book, we use an event-based system, where events triggered by player actions trigger API calls. For example, when a player starts a new level, a function call is sent to the API that describes the "begin level" event. Event descriptions follow this format: API_Call (event category, event action, location in the game, optional meta-data). The Event category allows us to group events into different categories, which makes it possible for example to differentiate between events that happen in the game's main menu and during gameplay. The event action describes the player action that triggers the event. For example, if a player clicks a "begin" button to start a level, the event action is "click." Other data added to each event describes the event further, including: location the event occurred, time of day the event occurred, the player's ID number, which game the event took place in, etc. Events are stored in the MySQL database allowing us to create queries that search, filter and aggregate the events to construct features for analysis.

Our variables included frequency, duration, achievements, and player choices as totals, averages, or percentages

The kinds of gameplay features we constructed can be classified into four different forms: frequency, duration, achievements, and player choices. Frequency refers to the number of times that a player performed certain action, e.g., number of shots, number of hints used, or number of upgrades. Duration refers to how much time players spent on a behavior, such as milliseconds before taking a first shot or time spent viewing a tutorial. Achievements refer whether and how well players succeed at game goals and subgoals such as completing a level or winning a round. Player choices refer to decisions players make such as selecting a difficulty level, choosing a particular lawyer, or choosing a type of weapon.

Frequency, duration, and achievement features can be collected as totals, percentages or averages. They can be collected to reflect overall gameplay or a specific subset of gameplay. For example, we could measure overall number of cases won or overall percent of cases won. These are overall measures of achievement. Alternatively, we could measure the number or percent of cases won in 10 min of play. Or we could measure number and percent of cases won by the end of level 2, regardless of how long it took each player to complete level 2.

Here is a typical example conundrum: potential clients in *Do I Have a Right* approach the player's law firm asking to be represented in court. The player must decide whether or not the client's complaint has a legal basis in the Bill of Rights. If they accept a client, they bring the case to court, and either win or lose the case. But does it matter that player X lost two cases and won ten cases, whereas player Y lost 6 cases and won 14 cases? We can decide that cases won are a measure of performance. Player X's performance is worse than Player Y's performance if we

look only at cases won. Player X's performance is better than Player Y's performance if we only look at cases lost. Player X has a better ratio of cases lost to cases won (20%) than Player Y (43%), so from that perspective Player X performed better.

We could decide that the "cases lost to cases won" ratio is a good performance metric. The ratio lets us make comparisons between players, regardless of how many days they have played. *Do I Have a Right* divides play into days in court, rather than levels. The first day in court is the easiest, with one lawyer to control, a handful of clients, and only one bill of rights amendment to deal with. All of these factors become increasingly complex in later days in court. It turns out that Player Y has completed 3 days in court and Player X only 2 days. That means Player Y has had more opportunities to encounter more new cases, and the difficulty of the cases increases as play continues. Therefore, we decided to compute total cases won and total case lost at the end of day four and compare cases won instead of the ratio of cases lost to cases won. Making these kinds of choices involves conceptualization, looking closely at player data and a firm domain knowledge.

Duration of play complicates metric definition. Enforcing duration of play is conducive to comparable metrics

Our first attempt at examining mindset and gameplay in *Do I Have a Right* involved 126 undergraduates who volunteered in order to earn extra credit. Participants completed a pretest that included the same questions Dweck uses to measure mindset and a second set of similar questions asking specifically about gaming mindset. They were asked to play *Do I Have a Right* for 10 min, and then to click CONTINUE to complete a post test.

Our initial data collection system did not enforce the request to play for at least 4 days (a little less than 10 min). Participants could opt to stop playing and continue to the survey any time after they started the game. We discovered that only 43% of participants actually played for at least 4 days. Fully 28% played for less than one courtroom day. (See Fig. 32.7) We revised the system to force study participants to play for at least 10 min, enough to complete 4 or more days in court. After 10 min of play, a CONTINUE button appeared. Participants could continue playing for as long as they wanted, but had to go on to the post survey to reach the extra credit request page. This time, everyone who finished the study completed at least 4 days in court within the minimum 10 min of play. Players play at different speeds, so to optimize player comparisons, we computed most of our metrics based on player accomplishments up to the end of day 4 in court, rather than based on time elapsed. Having at least 4 court days of data allowed us to construct and compare a rich set of metrics, described later in this case study.

This detour into specific complexities of feature construction illustrates some of the issues that arise when the intent is to compare across players. We return now to our variable creation process. We played *Do I Have a Right* many times, thinking about ways to represent key gameplay metrics related to mindset. With the theory of mindset in mind, we tried to operationalize performance and failure, feedback and

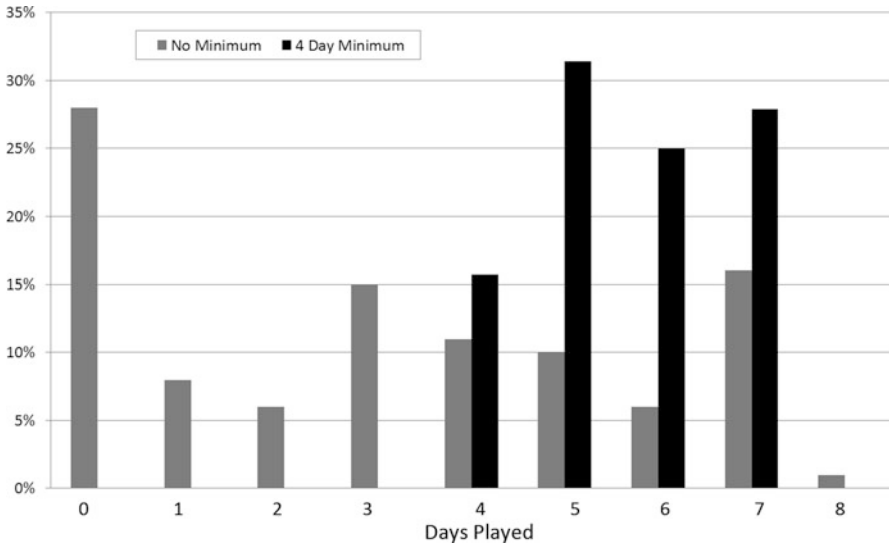


Fig. 32.7 Frequency distribution of commitment before and after enforcing 4 days virtual gameplay minimum

response to failure, and challenge-seeking. Players with a fixed gaming mindset were expected to avoid hard challenges, to fail to learn from failing, and therefore to fail more often and perform worse than players with a growth gaming mindset.

The data we will use to demonstrate our metric definition for mindsets comes from an experiment we conducted in 2010 and 2011. We recruited 123 university students from classrooms at a major Midwestern university. The students were asked to login to our experiment website, where they were asked to fill out a questionnaire that included questions to measure gaming mindset. Then the students were instructed and required by the system to play *Do I Have a Right* for duration of at least 10 min.

32.3.2.2 Gaming Mindset

Gaming mindset is measured using just four survey questions.

Gaming mindset was measured in the pre-game survey and will be used to compare fixed and growth mindset players' gameplay. Lee et al. (2012) showed that the gaming mindset is a better predictor of gameplay behavior than is general mindset. This finding is consistent with Dweck's (2006) expectation that we have different mindsets in different domains. The gaming mindset was calculated based on the following four questions on the pre-game survey, using a six point Strongly Agree to Strongly Disagree scale:

1. You have a certain gaming abilities and you really can't do much to change them.
2. Your gaming abilities are something about you that you can't change very much.
3. To be honest, you can't really change how good a gamer you are.
4. You can learn new games, but you can't really change your basic gaming abilities.

To calculate the gaming mindset, the responses were averaged. Therefore, the range of possible values was between 1 and 6. We reverse-coded the scale so that higher scores reflected more growth mindset. Those with a score between 1 and 3 were classified as fixed gaming mindset. Those with a score between 4 and 6 were classified as growth gaming mindset. Values between 3.01 and 3.99 were classified as ambiguous, and were omitted from further analysis. In our sample of 123 undergraduates, 29% had a fixed gaming mindset, 54% had a growth gaming mindset, and 17% were ambiguous. Mindset researchers typically omit those classified as having an ambiguous mindset to allow for clear comparisons between fixed and growth mindsets (for example, Mangels et al. 2006). This approach parallels how Dweck (2000, 2006) operationalizes mindset.

32.3.2.3 Performance Feedback

Performance feedback refers to the feedback that players get about their success or failure during and between rounds of play. Often the outcome is communicated immediately following a player action. Many games also have some form of summary performance feedback after each level. In *Do I Have a Right*, an immediate performance feedback is given after every correct or incorrect matches between client and lawyers. A more detailed feedback explaining why the choices are correct or incorrect is provided after each “day”.

Fixed mindset players won significantly fewer cases overall and they won a smaller proportion of the cases they brought to trial.

Players with a fixed gaming mindset hate to fail. When failure happens, they experience negative emotions and pay attention to their wrongness rather than thinking about how to play better next time. Although players with a fixed mindset try harder to avoid failure, they probably end up failing more than those with a growth mindset because they fail to learn from failing. Because failure is so critical to mindset, we considered three possible kinds of metrics to measure failure in *Do I Have a Right* and to see which is most appropriate for detecting mindset performance differences. First, we look at the raw number of cases won in 4 virtual days in court and compare its utility to that of the percent of cases won. The first column of Table 32.6 shows the performance of all players combined, including fixed,

Table 32.6 Number and percent of cases won

# Cases won	All	Fixed mindset	Growth mindset
1–4	24%	37%	18%
5–8	51%	51%	51%
9–17	24%	12%	31%
% cases won			
= < 75%	28%	47%	18%
76–99%	32%	26%	35%
100%	36%	26%	41%
n	149	43	80

Table 32.7 Average number and percent of cases won by gaming mindset

Performance	Fixed mindset	Growth mindset	t	df	p
# Cases won	5.8	<7.2	-1.375	121	.011
% Cases won	78%	<87%	-2.764	121	.007
n	43	80			

growth, and ambiguous mindset players. The second column shows only fixed mindset data. The third column shows only growth mindset data. We see that about one fourth of all players in our sample won between one and four cases; about half won five to eight cases; and another fourth won 9–18 cases, all in the first 4 days in court.

Looking only at cases won ignores cases lost. If instead we look at percent of cases won, 36% of all players won every case they tried; 32% won 76–99% of cases; and 28% won 75% or fewer of the cases they tried. Fixed gaming mindset players won significantly fewer cases ($\chi^2(2, N = 123) = 8.869, p = .012$): only 12 won 9–17 cases compared to 31% of growth gaming mindset players. Fixed gaming mindset players also won a smaller percentage of the cases they tried ($\chi^2(2, N = 123) = 6.021, p = .049$): 47% of fixed mindset players won 75% or less of their cases, compared to 18% of growth mindset players. Table 32.7 shows mean comparisons of those same metrics. Both are significantly different by mindset, using t-tests of statistical significance. The cases won metric focuses attention on winning. The percent of cases won metric focuses attention on failure rate. On both measures, fixed mindset players fared worse.

Next, we look more widely at failure in the game. Players receive feedback in the game about whether they correctly accept or reject each client. Therefore correct accept and correct rejects are a more detailed form of performance feedback than cases won. We look at the total number of incorrect rejects, total incorrect accepts, and the percent of failed client decisions. Incorrect rejects were less common than incorrect accepts. More than half of *Do I Have a Right* players never incorrectly rejected a potential client; only 15% incorrectly rejected two or more clients. On the other hand, only about one third of players never incorrectly accepted a client. When the two forms of error are combined, only 20% of players maintained a perfect decision record.

Table 32.8 Average incorrect reject, incorrect accept, and total incorrect by gaming mindset

Performance	Fixed mindset	Growth mindset	t	df	p
# Incorrect rejects	0.79	0.53			ns
# Incorrect accepts	1.56	>1.05	2.088	121	.039
Combined # incorrect	2.3	>1.6	2.620	121	.010
% of clients combined accept and reject incorrect	15.7%	>9.9%	2.893	121	.005
n	43	80			

Takeaway: fixed mindset players made incorrect decisions for significantly more of the potential clients they encountered.

Table 32.8 presents fixed and growth mindset mean comparisons. The number of incorrect rejections was not significantly different by mindset. Fixed mindset players committed significantly more incorrect accepts than growth mindset players (1.56 compared to 1.05) and significantly more combined incorrect choices (2.3 compared to 1.6). The number of potential clients a player encounters in the game depends upon choices they make during play, even if they have played for the same number of days. For this metric, the percent of incorrect client decisions is even more different than raw numbers: fixed mindset players made incorrect decisions for 15.7% of the potential clients they encountered, compared to 9.9% for growth mindset players.

Takeaway: almost all of these failure metrics were significantly worse for fixed mindset players. The percentage metrics revealed stronger differences than the raw numbers.

32.3.2.4 Attention to Feedback

Failing in games is often seen as a chance for players to learn (Gee 2007). Failure-related feedback comes in two forms, performance outcome feedback and learning feedback. Performance outcome feedback, discussed in the previous section, tells players whether they have succeeded or failed, or sometimes how well they have performed. Learning feedback, on the other hand, tells players how they succeeded or why they failed; this kind of feedback usually includes more detailed explanations that players can learn from to avoid mistakes in future gameplay.

Players with a growth gaming mindset spent more time on performance and learning feedback.

Table 32.9 Mindset feedback comparisons

Feedback/response to failure	Fixed mindset	Growth mindset	t	df	p
Seconds in amendment screen	10	16		121	ns
# Headlines read	1.77	<2.58	-2.001	121	.048
Seconds between day 1 and day 2*	66	<109	-2.965	62	.004
*n=20, 44	43	80			

The first two analyses were conducted with data collected in May 2011 with n=123, the third analysis (marked with * in the table) introduced a new variable in a dataset collected in April 2012 with n=64

Attention to learning feedback and delayed performance outcome feedback can be measured by frequency of accessing feedback descriptions or duration of time spent on those screens. Attention to instant performance outcome feedback is often unavoidable. Attention to learning feedback requires going to or staying on a feedback screen and also, actually absorbing the information. Gameplay metrics can only measure the former. We know from other research that fixed mindset players do not absorb learning feedback well, even when they are exposed to it. When exposure to that feedback is voluntary, do fixed mindset players avail themselves of the opportunity?

The most important form of learning performance feedback in *Do I Have a Right* is reading headlines between days in court. The game offers virtual newspaper screens, one per case tried, with headlines and explanations of the case. Players can skip the headlines entirely or read as many as they choose for as long as they wish before starting the next day in court. Table 32.9 shows comparisons between fixed and growth gaming mindset players. Twenty-seven percent of players did not read any of the between screens headlines, which recapped the cases of the day and, had they lost the case, would have been an opportunity to read the story and see why they were wrong. Twenty-two percent read only one headline; 12% read two; 11% read three; 14% read four; 7% read five, and another 7% read six to nine headlines.

Reporting just the number of headlines read metric masks how long players spent reading headlines. To be counted as read, a headline could be viewed for as little as 1 s. An alternative measure of attention to learning feedback is the time spent between days in court. For the 2010 subset of data, we computed time spent between day 1 and day 2. About one fourth of players spent less than 1 min between days in court, looking at learning and performance feedback; 23% spent between 1 and 1.5 min; another fourth spent 1.5–2 min; and 29% spent more than 2 min.

Players with a growth gaming mindset spent more time on performance and learning feedback, and read more headlines than did players with a fixed mindset.

Table 32.10 Mindset upgrades comparisons

Challenge-seeking	Fixed mindset	Growth mindset	t	df	p
# Desk upgrades	0.84	<1.45	-1.981	121	.050
# Amenity upgrades	0.20	0.20			ns
# Ad upgrades	0.37	<0.75	-2.279	121	.024
Total upgrades	1.42	<2.41	-2.060	121	.042
n	43	80			

32.3.2.5 Challenge-Seeking

Challenge-seeking relates to Dweck's theory of mindset (2006), which suggests that some individuals welcome hard challenges and others avoid failure. They find easy challenges boring and are resilient in the face of failure because they believe in their capacity to learn and improve. We developed metrics to try to track challenge-seeking play. A complicating factor is variability across player skills. What is hard for one player may be easy for another. Seeking challenge may come from choosing higher difficulty settings (in games that offer that choice) or from advancing/leveling up.

Players with a growth gaming mindset were significantly more likely to purchase upgrades that made gameplay more challenging.

Since *Do I Have a Right* does not explicitly offer players the choice of difficulty, we measured challenge by the kinds of upgrades players purchased in the game. Three kinds of upgrades were possible: desk upgrades, amenity upgrades, and advertisement upgrades. Desk upgrades increase the speed of lawyers, or the variety of amendments that a lawyer can handle. Thus, they increase the complexity of gameplay because the player juggles more cases and rights, but also enables more successes. Advertisements increase the number of clients that players must handle in each round, increasing the challenge. Amenity upgrades make the office more pleasant, reducing waiting clients' impatience. Amenity upgrades make the game easier to play.

Table 32.10 presents mindset comparisons. Players with a growth gaming mindset were significantly more likely to use desk upgrades and ad upgrades, both of which are consistent with greater challenge-seeking.

32.3.3 Metric and Design Implications

We have demonstrated the importance of controlling for difficulty level when comparing across players. We have explored the usefulness of raw totals versus percentages, and found some utility in each approach. The metrics we developed, on

failure, performance and learning feedback, and challenge-seeking show a range of player behaviors along each measure. The mindset comparisons confirm that more of the players who failed, and who ignored learning feedback, and who sought lesser challenges, had a fixed mindset, meaning that they viewed failure in a game as a negative assessment of their abilities rather than an opportunity to learn and improve. In addition to predicting gameplay, mindset theory gives us insight into how those players react to failure.

The data on time spent on feedback between day 1 and day 2 is particularly intriguing as a means of detecting players in need of remediation. There may be ways to catch players who are at risk of failing early in a game and provide them with support to hopefully keep them intrigued by the game long enough to where failure then becomes a learning experience instead of an exit point.

32.4 Conclusions and Next Steps

- We have shown two case studies where informative gameplay metrics were constructed to measure conceptually meaningful player behaviors. Four general types of game metrics were discussed and compared: frequency, duration, achievements, and player choices. Some metrics can be used as theoretically meaningful indicators alone, others need to be combined. These different types of metrics can be used to validate each other.
- Pre-testing the metrics in combination with a survey can help to further validate and make informed choices as the measures are refined. Our work benefitted from measuring key constructs in more than one way, comparing results, and being able to choose the most meaningful metric. We showed that controlling for level and duration of play is critical when designing metrics for cross-player comparisons. Enforcing a minimum amount of play (either based on time or an achievement milestone) facilitates this approach. Specifying a timeframe in the game or fixed amount of play-time within which variables are computed is essential for each metric.
- For learning games, the built-in achievement system (winning) is not the most informative indicator of whether a player learns from the game. Instead, distinguishing characteristics of play behavior that is likely associated with learning include how players approach a goal and what they do after failure.
- In brain games, tolerance of failure should be encouraged so that players take on hard enough challenges. Although some players in our study did pursue hard challenges, many did not. Brain games, which adapt to player performance to offer just enough new challenge, would be better for cognitive gain than self-selected challenge for many players.
- The case studies demonstrate the value of considering optimal challenge and gaming mindset during game design. Our next research steps will be to measure intended serious game outcomes (cognitive gain and learning) test how well the metrics described in this chapter predict those outcomes.

For readers interested in learning more about mindset, we recommend Dweck's books on the topic (Dweck 2000, 2006), two brain scan studies presenting neural evidence of mindset (Mangels et al. 2006; Moser et al. 2011a) and our own recent mindset research (Lee et al. 2012).

About the Authors

Carrie Heeter is a Professor in the Department of Telecommunication, Information Studies, and Media at Michigan State University, Program Director for MSU's fully online graduate certificate in serious games, Creative Director of Teaching and Learning Design and Technology Services, and a Principal in the GEL (Games for Entertainment and Learning) Lab. She co-founded and teaches in the MSU serious game design M.A. specialization. Heeter co-edited *Beyond Barbie and Mortal Kombat: New perspectives in gender and gaming*. She designs and studies meaningful applications of emerging media. Over the last two decades her interactive designs have won more than 50 awards including Discover Magazine's software innovation of the year. She is currently working on games to create a more informed, engaged citizenry and to facilitate deliberation and discussion about complex socio-scientific issues. Her most recent game, *DNA Roulette*, an experiential exploration of Direct to Consumer (DTC) genetic testing, was awarded Most Innovative Game at Meaningful Play, 2012.

Yu-Hao Lee is a Ph.D. student in Media & Information Studies at Michigan State University. His research revolves around digital games and motivation, specifically using games to facilitate environmental and civic action. He has published research on the virtual economy in Massively Multiplayer Online Games (MMOs), using digital games to address digital divide, and digital games for learning. He has also worked as a journalist covering environmental and political issues in Taiwan, in which he has won awards for both his academic and news work.

Ben Medler is currently with the Chief Creative Office of Electronic Arts. Prior to that he was a Ph.D. student at the Georgia Institute of Technology. His research revolves around using information visualization techniques to build game analytic tools for both game designers and players. He has built visual game analytic tools for Electronic Arts and has developed a proprietary analytic system for capturing data from Flash-based games used for research studies. Ben is pursuing research in a number of related areas using his analytic tools including: understanding differences in player behavior, combining recommendation systems with analytics, and researching the ethics behind gathering player gameplay data. Additionally, he has authored and present publications on building adaptive game systems, interpreting conflicts in games and linking improvisation with role-playing.

Dr. Brian Magerko is an Assistant Professor of Digital Media in the School of Literature, Communication, and Culture at the Georgia Institute of Technology. He is director of the Adaptive Digital Media (ADAM) Lab, which explores how to

use artificial intelligence to create digital media experiences that tailor themselves to the individuals that use them, and a principal member of the Experimental Game Lab. Dr. Magerko has published numerous articles on serious games projects that employ artificial intelligence to adapt narrative and learning content to individual players. He teaches in the Computational Media program, a joint offering between the College of Computing and Digital Media, and in the Digital Media graduate program.

Acknowledgments Jennifer Mangels, Associate Professor of Psychology at Baruch College & Graduate Center, City University of New York, served as a consultant.

Carrie Cole, serious game design, digital media arts and technology, and HCI MA student at Michigan State University, contributed to manuscript revisions.

Hero Interactive LCC, Filament Games, iCivics, and the Michigan State University GEL Lab (games for entertainment and learning) provided the games in this study.

This research is partially supported by grant #0943064 from the National Science Foundation. The opinions expressed are those of the authors and do not represent Michigan State University, Georgia Institute of Technology, or the National Science Foundation.

References

- Aldrich, C. (2010). *Flow states in sim design: When to use it, and when to avoid it*. Retrieved January 29, 2012 from <http://www.clarkaldrichdesigns.com/2010/06/one-way-to-measure-sim-for-flow-how-far.html>.
- Beume, N., Danielsiek, H., Eichhorn, C., Naujoks, B., Preuss, M., Stiller, K., & Wessing, S. (2008). Measuring flow as concept for detecting game fun in the Pac-Man game. *Evolutionary Computation, 2008, CEC 2008, (IEEE World Congress on Computational Intelligence)*, pp. 3448–3455.
- Blackwell, L., Trzesniewski, K., & Dweck, C. S. (2007). Implicit theories of intelligence predict achievement across an adolescent transition: A longitudinal study and an intervention. *Child Development, 78*, 246–263.
- Bozoki, A., Radovanovic, M., Winn, B., Heeter, C., & Anthony, J. (under review). Estimated effects of a computer-based cognitive exercise program on age-related cognitive decline. *Archives of Gerontology and Geriatrics*.
- Csikszentmihályi, M. (1990). *Flow: The psychology of optimal experience*. New York: Harper and Row.
- Dweck, C. (2000). *Self-theories: Their role in motivation, personality, and development. Essays in social psychology*. Philadelphia: Taylor & Francis.
- Dweck, C. (2006). *Mindset: The new psychology of success*. New York: Random House.
- Fernandez, A. (2007a). *Brain exercise and brain health FAQs*. Retrieved September 5, 2008, from <http://www.sharpbrains.com/blog/2007/04/03/brain-exercise-faqs/>.
- Fernandez, A. (2007b). Build your cognitive reserve – Yaakov Stern. Retrieved September 5, 2008, from <http://www.sharpbrains.com/blog/2007/07/23/build-your-cognitive-reserve-yaakov-stern/>.
- Filament Games. (2010, May 13). *Evaluation ruling: Our courts works 13 May 2010*. Retrieved January 29, 2012, from <http://www.filamentgames.com/projects/argument-wars>.
- Gee, J. P. (2007). In J. P. Gee (Ed.), *Games and learning: Issues, perils and potentials. Good video games and good learning: Collected essays on video games, learning and literacy: New literacies and digital epistemologies* (pp. 129–174). New York: Palgrave/Macmillan.
- Happy Neuron. (2012a). *Brain game collection*. Retrieved January 29, 2012. <http://www.happy-neuron.com/tools/>.

- Happy Neuron. (2012b). *How happy neuron works*. Retrieved January 30, 2012, from <http://www.happy-neuron.com/about/how-it-works>.
- Hardy, J., & Scanlon, M. (2009). *The science behind lumosity*. Accessed Jan 2012, from <http://www.lumosity.com/blog/the-science-behind-lumosity/>.
- Heeter, C., Winn, B., Winn, J., & Bozoki, A. (2008). *The challenge of challenge: Avoiding and embracing difficulty in a memory game*. Presented at Meaningful Play Conference, East Lansing, MI
- Lee, Y., Heeter, C., Medler, B., & Magerko, B. (2012a). Gaming mindsets: Implicit theories in serious game learning. *Cyberpsychology, Behavior and Social Networking*, 15, 190–194.
- Lumosity. (2011). *B.P.I. in a nutshell*. Retrieved January 30, 2012, from <http://www.lumosity.com/blog/bpi-in-a-nutshell/>.
- Mahncke, H., Bronstone, A., & Merzenich, M. (2006). Brain plasticity and functional losses in the aged: Scientific bases for a novel intervention. In A. Moller (Ed.), *Progress in brain research* (Vol. 157). Amsterdam/New York: Elsevier.
- Mangels, J., Butterfield, B., Lamb, J., Good, C., & Dweck, C. S. (2006). Why do beliefs about intelligence influence learning success? A social cognitive neuroscience model. *Social Cognitive and Affective Neuroscience*, 1(2), 75–86.
- Moser, J., Schroeder, H., Heeter, C., & Lee, Y. (2011a). Mind your errors: Evidence for a neural mechanism linking growth mindset to adaptive post-error adjustments. *Psychological Science*, 22, 1484–1489.
- Owen, A., Hampshire, A., Grahn, J., Stenton, R., Dajani, S., Burns, A., Howard, R., & Ballard, C. (2010). Putting brain training to the test. *Nature*, 465(7299), 775–778.
- Salen, K., & Zimmerman, E. (2004). *Rules of play*. Boston: MIT Press.
- Scanlon, M., Drescher, D., & Sarkar, K. (2006, October). *Improvement of visual attention through a web-based training program*. Poster presented at Society for Neurosciences Annual Meeting in Atlanta, GA
- Scanlon, M., Drescher, D., & Sarkar, K. (2007a, January). *Improvement of visual attention through a web-based training program*. Poster presented at Bay Area Neuroscience Gathering in San Francisco, CA
- Scanlon, M., Drescher, D., & Sarkar, K. (2007b). *Improvement of visual attention and working memory through a web-based cognitive training program*. A Lumos Labs White Paper
- Sluijs, E. M., Kok, G. J., & van der Zee, J. (1993). Correlates of exercise compliance in physical therapy. *Physical Therapy*, 73(11), 771–782.
- Willis, S., Tennstedt, S., Mariske, M., Ball, K., Elias, J., Koepke, K., Morris, J., Rebok, G., Unverzagt, F., Stoddard, A., & Wright, E. (2006). Long-term effects of cognitive training on everyday functional outcomes in older adults. *Journal of the American Medical Association (JAMA)*, 296(23), 2805–2814.

Related Research on In-Game Metrics

- Heeter, C. (2008). Playstyles and learning styles. In R. Ferdig (Ed.), *Handbook of research on effective electronic gaming in education*. Hershey: IGI Global.
- Heeter, C., & Winn, B. (2005, October). Gender, playstyle, and learning: Constructing in-game measures of playstyle. In *FuturePlay Proceedings*, East Lansing.
- Heeter, C., & Winn, B. (2008). Implications of gender, player type and learning strategies for the design of games for learning. In *Beyond Barbie to Mortal Combat: New perspectives on games, gender, and computing*. Cambridge, MA: MIT Press.
- Heeter, C., Winn, B., Winn, J., & Bozoki, A. (2008, October). The challenge of challenge: Avoiding and embracing difficulty in a memory game. In *Proceedings of the meaningful play conference*, East Lansing

- Heeter, C., Lee, Y., Medler, B., & Magerko, B. (2011a) Beyond player types: Gaming achievement goal. Vancouver: SIGGRAPH, August. <http://gel.msu.edu/carrie/publications/SIGGRAPH2011-paper.pdf>
- Heeter, C., Lee, Y., Magerko, B., & Medler, B. (2011c). Impacts of forced serious game play on vulnerable subgroups. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, 3(3), 34–53.
- Lee, Y., Heeter, C., Magerko, B., & Medler, B. (2012b). Gaming mindsets: Implicit theories in serious game learning. *Cyberpsychology, Behavior, and Social Networks*, 15(4), 190–194.
- Moser, J., Schroeder, H., Heeter, C., & Lee, Y. (2011b). Mind your errors: Evidence for a neural mechanism linking growth mindset to adaptive post-error adjustments. *Psychological Science*, 22, 1484–1489.

Chapter 33

Interview with Simon Egenfeldt Nielsen from Serious Games Interactive

Alessandro Canossa

Serious Games Interactive (SGI) started in 2006 as a research spinoff. The company was founded in cooperation with Unity Technologies as a technology provider with the ambition of making the world's best serious games by harnessing the potential and design principles of real AAA titles. The company received considerable critical praise both from educators and students. In 2010 it was awarded a BETT Award in the category of "Secondary, FE & Skills Digital Content" for its flagship title *Global Conflicts: Palestine* (Gamers Gate, Manifesto Games & Macgamestore, 2007). The game *Playing History – The Plague* (Gamers Gate, 2009) was also awarded first place in the category of professional games (budget 40,000+ Euros) in the 1st European Best Learning Game competition.

Since its inception the company has developed and completed more than 30 projects that show how games can be used for much more than entertainment in a variety of fields.

Simon Egenfeldt Nielsen is both founder and managing director of the company; as such he attempts to keep involved in all aspects of production. Over the years he has been involved with both research and industry. In 2005 he obtained a PhD from the IT University of Copenhagen on the educational use of computer games, after that he led a 2 years long research project within the same field and eventually he decided to found Serious Games Interactive.

Q: Does SGI make use of game telemetry systems?

S: It is a real shame to admit that we are not really taking advantage of these systems. We have ambitions to streamline production processes by monitoring both performance and process metrics (see Chap. 2), unfortunately we haven't had the

A. Canossa, Ph.D. (✉)
College of Arts, Media and Design, Northeastern University,
Boston, MA, USA

Center for Computer Games Research, IT University of Copenhagen,
Copenhagen, Denmark
e-mail: a.canossa@neu.edu

chance to plan a full-fledged deployment of such tools. On the other hand we have started looking into monitoring and profiling player behavior with the intent of creating adaptive game experiences. This is the area with the most interesting potential: telemetry systems to support adaptive technologies. It is tricky, though. There are a few products in the learning game space, which have implemented adaptive technologies in a very simple if not trivial manner. For example the systems monitors player performance in answering questions and automatically adjusts difficulty to Easy, Medium or Hard, which are pre-defined sets of questions with varying difficulty. Even if this solution was fairly simple and straightforward, it has been hyped considerably. There are quite a few companies at the moment adopting similar solutions. We tried Yogi Play¹, a solution that turns mobile phones and tablets into smart learning devices by embedding different learning and game objectives into an API that different developers can include in their applications. The goal is to provide parents with meaningful insights related to their child's learning progress. The system provides children with personalized achievements and educators with recommendations based on a child's specific learning needs. The idea in principle was good so we implemented it, but after release we have been hammered by review sites and decided to remove the API from our game. The problem lies with the fact that a log in screen keeps popping up during play, and that is not very user friendly, especially when dealing with children. The intention is good, since Yogi Play attempts to track a multitude of users across a variety of games from different providers, but the implementation was not satisfactory for a product intended for children. The core idea to adapt a game according to player behavior is an obvious advantage, but too often the implementation is reduced to a set of monitoring tools for parents and teachers that allow controlling what the kids are doing, rather than engaging in more interesting challenges. For example it would be interesting to weave player behavior in the main mechanics of play. I don't think that parents spend a lot of time drilling through the reports on their children's play and learning behavior, maybe it's more pertinent for educators, but I personally would rather use the technology to provide meaningful experiences.

Q: What is the state of the art regarding metrics tracking for learning games?

Simon: The Norwegian game *3-7*² (Egmont Serieforlaget, 2012) is a pretty good example, it provides a cross-disciplinary learning environment for several subjects: math, history, literature, etc. The game tracks player progress and skills and provides a very basic form for adaptation: easier or more difficult scenarios are chosen according to the number of errors committed. Another example is the Danish *ABCiTY*³ (ABCiTY, 2011), an adaptive application to teach children how to read and write. Even in this application the task's difficulty level is adjusted to the child's proficiency. On the same line, *Manga High*⁴ (Blue Duck Education, 2009) offers

¹ <http://www.yogiplay.com/>

² <http://3til7.klikk.no>

³ <http://www.abcity.dk>

⁴ <http://www.mangahigh.com>

game-based math teaching that dynamically adapts the difficulty to the ability of the student in order to make the user experience engaging, entertaining and personal. All of these applications rely heavily on behavior tracking to adapt the difficulty of learning tasks, but the content stays the same. Several venture capital funds and business angels have expressed interest in metrics-based adaptive learning environments but until now all that I have seen is rather straightforward difficulty adjustment systems that select one of several pre-designed sets of tasks; even procedural assembling of sets has not been tried yet, at least to my knowledge. True innovation on this field is beyond the resources of the small developers that have tackled the challenge until now.

Q: Has SGI attempted to procedurally adapt content or difficulty levels based on metrics tracking?

Simon: To a certain degree, we are trying it with two projects funded by the European Community: Siren and Games for Health, both carried out within large consortia involving several industry and academic partners. SIREN⁵ (Social games for conflict RESolution based on natural iNteraction) is a game that takes a stab at conflict resolution, taking advantage of recent advances in serious games, social networks, computational intelligence and emotional modeling to create uniquely motivating and educating games that can help shape how children think about and handle conflict. The software, developed in cooperation with the IT University of Copenhagen among others, will be able to automatically generate conflict scenarios that fit the teaching needs of particular groups of children with varying cultural background, maturity, and technical expertise, and the desired learning outcomes as specified by a teacher. Games for Health⁶ investigates if and how digital gaming can contribute to creating new and improved therapies for veterans diagnosed with post-traumatic stress disorder (PTSD). By monitoring player behavior and bio-physiological feedback (stress levels), the simulation can be tailored to mimic the experiences that caused the soldier to develop PTSD and procedurally adapt different stressors, stressors are here intended as stimuli that are recognized as leading to manifestations of stress in war veterans and can range from sounds to light flashes, voices, music and visual stimuli. In both examples, metrics are gathered and used to adapt content in a way that goes beyond selecting pre-designed scenarios. Success for these two applications can only be guaranteed with granular and nuanced adaptation rather than pre-set, multiple choices. There is a dual challenge here: establishing relevant measures and creating modular designs able to adapt.

Q: What advice would you give to studios trying to adopt metrics-based adaptivity for content generation?

Simon: Definitely I would recommend not slapping a metrics-based adaptive solution on a nearly completed product, but investing in metrics definition and tracking while still designing the application; only in this way it is possible to create truly

⁵<http://sirenproject.eu/>

⁶<http://www.gamesforhealth.dk>

meaningful adaptive content. I am aware of how difficult that is, because in the earlier stages, all resources are already devoted to other aspects of production and it is particularly challenging to tackle issues for which very little in-house knowledge exists, like defining behavioral variables for adaptive procedural content generation. It is important to accept partial failures as learning experiences, the team needs to understand how to work with these technologies and how to best harness their potential; the implications are far reaching, involving everybody: programmers, game and level designers, artists and even management.

Q: What do you think can be achieved in the future by adopting metrics-based adaptive practices?

Simon: From the end user perspective, correctly implementing adaptive techniques implies reducing considerably frustration, which is the main hindrance towards engaging experiences and therefore learning. The low-hanging fruits of that process are obviously excellent tools for evaluation, where educators and parents alike can monitor children's progresses. In fact, with no need for expensive adaptive technologies, tracking players' behavior can provide a close account of a session. From a developer's perspective, adaptive approaches could potentially reduce the workload in producing extensive amounts of content, once the technology is mature enough.

Q: Do you have experiences in utilizing metrics also for user research or optimizing production processes?

Simon: The main challenge for utilizing metrics in user research is that SGI mostly develops single player campaign games and not permanent multiplayer worlds. Once our games are released, there are no updates or patches, since our clients do not pay for ongoing support. During development we do not have enough testers to reach a statistically significant mass to employ quantitative methods, therefore we cannot produce meaningful reports; we make use of traditional, qualitative user research methods (see Chap. 2). We are struggling at the moment to convince clients to invest in longer term support for their products, and that is based on the obvious benefits offered by telemetry systems for behavior analysis. It is just a matter of time before our clients grow and understand the need for this long tail, at that point they will be asking for it themselves. The first sign is coming from the Danish Ministry of Foreign Affairs, we have cooperated with them on several projects already and they seem keen on adopting telemetry to improve their products and assess the success rate, especially for the ongoing series of campaigns we have been developing together for the past 5 years: it's a series of virtual worlds where kids can explore life in developing countries. These games build on each other and evolve, constantly improving the design. In this case, analyzing player behavior makes perfect sense and the client understands the benefits involved even if it requires a slightly larger budget. We are therefore looking at which games are more popular, how often they are played, for how long, what do players do, etc. We intend to harvest this information and capitalize on it for the next iteration of the campaign. It's not very sophisticated but it fulfills our needs for now.

Part VIII

Metrics and Content Generation

This final part of the book deals with design-time metrics for level generation in deductive puzzle games. Metrics are used to provide:

- automated generation strategies for game spaces and challenges
- deductive estimation of challenges iterative and strategic depth solution to the challenges
- solutions to challenges that increase the perception of intelligence and personality behind level designs

The part includes only one chapter: Chapter [34](#) (*Metrics for Better Puzzles*), by Cameron Browne from the Imperial College London. The chapter builds a case for using metrics to generate content in puzzle games.

Chapter 34

Metrics for Better Puzzles

Cameron Browne

Take Away Points:

This chapter provides a case study of automated level design for a new solitaire puzzle.

1. It demonstrates the application of design-time metrics, for classifying levels and encouraging desirable properties such as symmetry and strategic depth.
2. It explores ways in which puzzle metrics may be harnessed to produce level designs that are more interesting for players.

34.1 Introduction

The current crop of smart phones and handheld game devices are the ideal platform for logic puzzles, which are enjoying a surge in popularity due to the mainstream success of titles, such as *Sudoku* and *Kakuro*. Such puzzles can be played easily on small screens without losing any of their appeal, and can provide a deep, engaging playing experience while being conveniently short and self-contained.

Figure 34.1 shows a logic puzzle game, called *Hour Maze*, recently developed for iOS devices by Cyberite Ltd. I have developed tools for automatically generating and solving *Hour Maze* levels and measuring key metrics; however, the ability to quickly generate large numbers of levels raises certain questions:

- How do we know which of these levels will interest players?
- How do we make levels more attractive and engaging?
- How do we fine-tune the search to focus on fewer, better designs?

C. Browne (✉)
Imperial College London, London, UK
e-mail: camb@doc.ic.ac.uk; cambolbro@googlemail.com

Fig. 34.1 Full size 12×12 *Hour Maze* in the iPad prototype



There has long been debate over the merits of computer-generated puzzles compared to those handcrafted by human experts. Japanese publisher Nikoli, the world’s foremost publisher of logic puzzles such as *Sudoku* and *Kakuro* (Nikoli 2001), is suspicious of automatically generated content and its potential to flood the market with inferior mass product. For example, Nikoli’s chief editor Nobuhiko Kanamoto (2001) observes that: “*Computer-generated Sudoku puzzles are lacking a vital ingredient that makes puzzles enjoyable - the sense of communication between solver and author.*”

The aim of this chapter is to investigate how metrics may be used to classify puzzles and help address perceived biases against computer-generated content, using *Hour Maze* as a test case. I stress that the term *metrics* in this chapter refers to design-time metrics for puzzle design, i.e. metrics developed and applied during the design phase of a games development, rather than (tele)metrics obtained from players at run-time, as described in the other chapters of this book.

This chapter begins by outlining some basic principles of puzzle design and introducing *Hour Maze* more fully. I then describe strategies for solving *Hour Maze* levels and a deductive search method that can quickly find a puzzle’s unique solution based on these strategies. Metrics for measuring key aspects of a given level of the game are described, namely symmetry and depth, and a method for automatically generating levels using these metrics is outlined. The final sections suggest how such puzzle metrics may be harnessed to produce more engaging puzzles.

34.2 Puzzles

Salen and Zimmerman (2004) define games as follows:

A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome (p.81).

Crawford (1984) describes puzzles as rule-based systems like games, in which the goal is to find a solution rather than beat an opponent. Similarly, Kim (2008) defines puzzles as a form of play that: (1) is fun, and (2) has a right answer. *Hour Maze* is a form of *solitaire puzzle*, i.e. a puzzle that involves a single player: the solver.

Rollings and Morris observe that all “all games have puzzles” (2004, p. 37), and make the distinction that puzzles posit *specific* problems, whereas games are systems that spawn *generic* problems. Thompson (2000) makes a similar observation about the intimate relationship between games and puzzles, in the context of abstract board games:

Every board position presents the player with the puzzle, ‘What is the best move?’, which in theory could be solved by logic alone. A good abstract game can therefore be thought of as a “family” of potentially interesting logic puzzles, and the play consists of each player posing such a puzzle to the other (p.1).

Similarly, solitaire puzzles can also be viewed as two-player games that provide a contest between the *designer* who creates the puzzle and the *player* who tries to solve it. The designer constitutes a “null” player, who may not be physically present for the contest, but whose wit and personality can be evident in the challenge that they set to the solver (if the level is well designed). It is this feeling of intelligence behind each level design that we are trying to capture.

The defining feature of puzzles is that each instance has a correct solution, and once that solution is found then that exact instance is no longer interesting to the player until they forget that solution (Trefay 2010). In other words, each instance of a puzzle has strictly finite *replayability* value. In the words of Jesse Schell, “a puzzle is a game with a *dominant strategy*”, that is, a strategy that will defeat it every time (2008).

In order to keep a puzzle interesting for the player, it is necessary to generate a large number of new instances that the player has not yet seen; enough to keep the player engaged as long as they remain interested in playing the puzzle. And if the puzzle is as popular and addictive as *Sudoku*, then that means a *lot* of instances are required to meet global demand.

One way to satisfy this demand for new puzzle instances is through *procedural content generation* (PCG), which is the use of computers to automatically generate content for defined systems, using rule- and search-based methods (Togelius et al. 2011). Such methods are indeed able to generate large amounts of new puzzle content quickly, ranging in difficulty as required; however, such generated content can seem somewhat sterile to players, and lack the creative flair that is evident in puzzles carefully handcrafted by human experts. The aim of this chapter is to look at ways in which aspects of this “human touch” might be incorporated into procedurally generated puzzle content.

34.2.1 Japanese Logic Puzzles

Hour Maze belongs to a class of solitaire puzzles known as *Japanese logic puzzles*, due to their popularity in that country. Nikoli, the foremost publisher of such puzzles, provides the definitive catalogue, including *Sudoku*, *Kakuro*, *Slitherlink*, and so on. These puzzles are characterized by the following qualities:

- Single player.
- Simple rules.
- Unique solution.
- Can be solved by deduction.
- Context free, i.e. universal symbols such as numbers, not letters or words.

Japanese logic puzzles may be described as *pure deduction puzzles*, as the unique solution to each puzzle may be determined by logic and deduction alone without the need for guesswork. Some planning ahead may be required to solve particularly difficult situations, but such planning should take the form of existence proofs that prove or eliminate choices rather than simply finding possible candidates and “trying them out”.

Nikoli designer Hiroshi Higashida describes puzzles as questions that challenge the player, require their deduction according to the rules, and do not depend on chance or actions from other players (Higashida 2010). By the same token, he points out that designers are also bound by the rules, and that these dictate the ultimate designs.

The history of Japanese logic puzzles is tied closely with the history of its most prominent example, *Sudoku*. *Sudoku* is based on a mathematical construction known as the *Latin square*, which is an $n \times n$ array is filled with symbols that occur exactly once in each row and column (Higgins 2007). Latin squares were studied by the mathematician Leonhard Euler in the eighteenth-century, and close precursors of *Sudoku* were proposed in the 1890s, but it is generally believed that the modern *Sudoku* puzzle in the form that we know today was first proposed by US puzzle hobbyist Howard Garns and published in *Dell Pencil Puzzles & Word Games* under the name *Number Place* in 1979 (Pegg 2005). Nikoli then released it in their catalogue of “culture independent” puzzles under the name *Sudoku* (meaning “single number”) in 1984.

However, it was not until 2004–2005 that *Sudoku* became an international sensation, when it starting appearing as a regular feature in UK newspapers on a mass scale. This was made possible by Wayne Gould’s computer programme *Poppacom Sudoku*, which was designed specifically to generate *Sudoku* content for the global market. Gould reported earnings from *Poppacom Sudoku* of over a million dollars in less than a year (La Monica 2006), and went on to become named as one of the “World’s Most Influential People” by *Time Magazine* in 2006.

The rise in popularity of similar Japanese logic puzzles, such as *Kakuro*, *Slitherlink*, and so on, can be traced to the rise of *Sudoku*. However, Japanese publisher Nikoli remain adamant that human-generated puzzle designs are superior to

those generated algorithmically, and remain distinct from most other publishers in the world by preferring not to release computer-generated designs, despite the proven convenience and cost-effectiveness of doing so.

34.2.2 Existing Approaches

There has been considerable research interest in Japanese logic puzzles, almost all of which focusses on complexity analyses, and automated methods for finding solutions and measuring strategic difficulty. For example, Kendall et al. (2008) provide a comparative complexity analysis between several Japanese logic puzzles and other types of puzzles, and show that they are *NP-Complete*, meaning that:

In general, it is much harder to find a solution to a problem than to recognize one when it is presented. (Cook 1984)

Automated methods proposed for solving logic puzzles include the following:

1. *Evolutionary Approaches*: Sato and Inoue (2010) describe the use of a *genetic algorithm* (GA) to solve *Sudoku* cases ranked as “super difficult” that proved problematic for other solution methods. They use a standard string-based representation for the puzzles, but define special crossover and mutation operators that preserve building blocks within the puzzles.
2. *Constraint Satisfaction Approaches*: Herting (2004) represents *Slitherlink* cases as constraint-based logic problems, and uses SAT (Satisfiability) Problem approach to find their solution.
3. *Binary Decision Diagrams*: Knuth (2011) uses structures called *binary decision diagrams* (BDDs) and *zero-suppressed decision diagrams* (BDDs) to prove solutions for given *Slitherlink* cases, and also for cases of his own variation *Skimperlink*.
4. *Monte Carlo Methods*: Cazenave describes the use of *Nested Monte Carlo* (NMC) search for achieving spectacular results in a number of solitaire puzzles including 16×16 *Sudoku* (2009).
5. *Tree-Based Methods*: Jing et al. (2009) describe a “branch and bound” *depth first search* (DFS) approach for a type of Japanese logic puzzle.

There therefore exist a number of approaches for automatically finding solutions to a given puzzle instance. Yato (2003) demonstrates that it is hard to determine whether a given solution is unique, for at least one type of Japanese logic puzzle.

Pelánek (2011) demonstrates how constraint satisfaction methods can also be used to measure the difficulty of *Sudoku* puzzles. He found that there were two sources of difficulty in a puzzle: the complexity of the individual steps (logical operations) and the structure of dependency among steps. Lee et al. (2008) went further to study the psychology of players in solving *Sudoku* cases over four levels of difficulty, and found phenomena that cannot be explained by current theories of deductive reasoning. They found that naïve individuals were able to acquire simple

deductive tactics in order to solve easy and mild *Sudoku* puzzles, making deductions about abstract matters remote from everyday life, and that those players able to move onto the more difficult puzzles were those able to make a strategic shift, and string together several of the learnt tactics into more complex strategies.

I believe that it is this diversity of strategies that makes a puzzle interesting. For example, players may become addicted to *Sudoku* when they first encounter it, but after a year or two the fascination begins to pall as they find themselves applying the same set of strategies to puzzle instance after puzzle instance. However, variants such as *Killer Sudoku*, which are equivalent in almost every respect, remain fascinating for year after year, puzzle after puzzle, as each puzzle instance has the scope to require the solver to devise new strategies that they have not encountered before. It is therefore desirable to allow the greatest possible scope for novel strategies in the design of any new puzzle type.

By contrast, there has been relatively little work on PCG approaches for logic puzzle levels. Methods have been proposed for the generation of *nonograms* or Picture-logic puzzles, which involve pictures that are revealed by deducing pixel locations (Ortiz-Garcia et al. 2007), and general logic puzzle levels using evolutionary techniques (Ashlock 2010). These approaches produce levels of desired difficulty, but do not address issues of aesthetics or player preference (apart from difficulty) in level design.

While some computer-generated content does make it to print, these PCG methods typically have a commercial motivation, as their purpose is to satisfy the constant global need for new content, and are therefore not publicly released. It is probably safe to say that no current PCG approach can produce puzzle designs as good or varied as those produced by the best human designers. However, we are starting to see board games of publishable quality being produced by automated systems based on aesthetic measurements (Browne and Maire 2010). Similar approaches may be applied to puzzle and level design.

34.3 Hour Maze

In this section, I describe *Hour Maze* in more detail. Figure 34.2 shows an example 7×7 *Hour Maze* level (left) and its solution (right). Note that this simple example is a miniature size with only four color sets for the sake of clarity; the full puzzle size is 12×12 and includes 12 color sets. Note that the Color sets in the figure are gray-scale for publication here, but are full Color in the actual game.

The goal of the game is to fill up the grid with numbers. The way this is done follows a set of simple rules, as follows: The player must fill the grid with hour values (numbers ranging from 1 to 12) and assign each one a color such that:

- There is exactly one number of each color.
- Numbers adjacent in the grid (i.e. not separated by a wall) must be sequential on the clockface. For example, 12 can only be adjacent to 1 and/or 11.
- Each color group forms a single connected set.

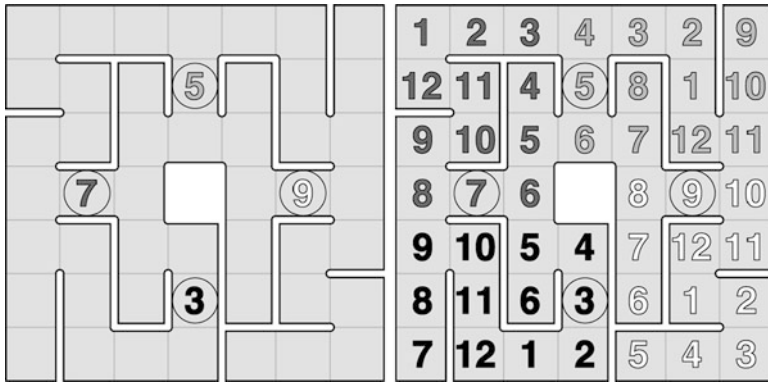
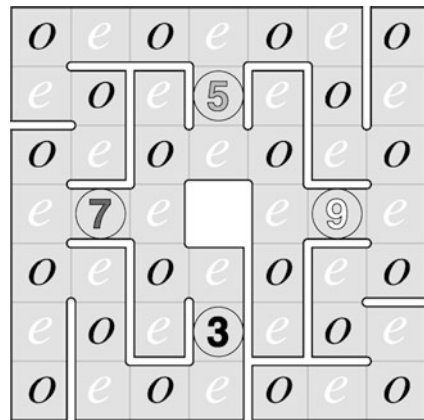


Fig. 34.2 Example level with solution

Fig. 34.3 Hints enforce a parity on each level



One number of each color is revealed as a *hint* to start the player off (circled). The numbers within each color need not start at 1, as long as they are sequential.

34.3.1 Definitions

We distinguish between the *puzzle* (the unique combination of rules that defines *Hour Maze*) and its *levels* (each specific instance of the puzzle). Each level consists of a:

- *Grid*: square grid, typically square in footprint.
- *Maze*: the wall pattern within the grid.
- *Hint set*: the number of each color revealed to the player (circled).
- *Solution*: the unique {*number, color*} pairs realized for each cell.

The square grid enforces a parity on each puzzle (see Fig. 34.3). The player can deduce from a given hint set whether each cell will contain an even (*e*) or odd (*o*) number. Each colored group of numbers 1...12 is described as the *hour set* or *color set* of color *c*.

34.3.2 History

The idea for *Hour Maze* was devised by puzzle designer Mike Reilly in 2007. The original incarnation of the puzzle used color to demarcate hour sets without regard for connectivity; color provided decoration and clarity but was not at that stage an integral part of the puzzle. Color was removed altogether for a black and white print run that presented levels as monochrome collections of H hour sets.

At this stage *Hour Maze* was not a pure deduction puzzle as each level supported multiple solutions and was typically solved by intuition, guesswork and backtracking. However, the notion of group color provided an obvious way to promote the puzzle to the pure deduction class with one simple rule; that each color set forms a single connected group. This additional rule was found to add strategic depth and forms the current rule set as implemented for the iOS prototype.

We considered also applying an additional *Sudoku*-style rule forbidding the same number to occur in any row or column. However, this was found to be unnecessary (the connected color rule adds sufficient depth) and would have had the unfortunate side-effect of placing *Hour Maze* as yet another *Sudoku* variant. As Higashida (2010) says, “the simpler the rule is, the better (p. 218).”

Higashida (2010) points out that the usual deductive rules were relaxed for the Nintendo DS implementation of his popular *Number Link* puzzle, by allowing multiple solutions for each level and imposing a time limit. Players were encouraged to replay each level in an effort to improve their best time, which introduces a new competitive element but means that *Number Link* DS is no longer a pure deduction puzzle. We decided not to follow this route for the iOS implementation of *Hour Maze* but to keep it as a pure deduction puzzle, for the sake of elegance and the added challenge to players.

34.3.3 Walkthrough

To give a taste of the puzzle, we now describe the steps that a player might take to solve the 7×7 level shown in Fig. 34.2. This small example can be solved using relatively few strategies, while full size 12×12 levels with 12-hour sets are typically harder and require more sophisticated strategies for a solution.

Firstly, observe that white is the only color that can reach the dead end *a* (Fig. 34.4, left) within 12 steps. This entire corridor must therefore be white (right).

Color sets cannot branch (explained shortly), so the white set must extend to the left of the hint 9 through cell *b* (Fig. 34.5, left). It cannot extend upwards from *b* as this would isolate corridor *c*, so the white set must extend to *c*. Cell *b* must be numbered either 8 or 10 due to the 9 hint, and is three steps away from the hint 5 so cell *b* must be 8; the ordering of the white group is therefore known (right).

Using similar logic, light grey is the only color that can reach dead end *e* and this set must extend to dead end *d*, as any other coloring would isolate at least one corridor (Fig. 34.6, left). The position of all light grey cells is therefore known and their number order determined by the nearby white hour set (right).

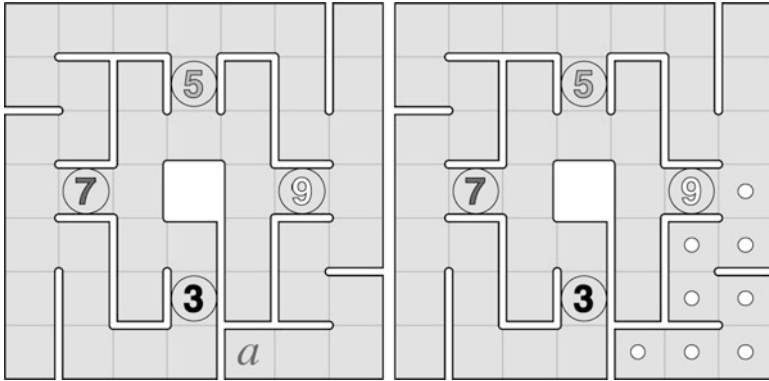


Fig. 34.4 Only white can reach cell *a*

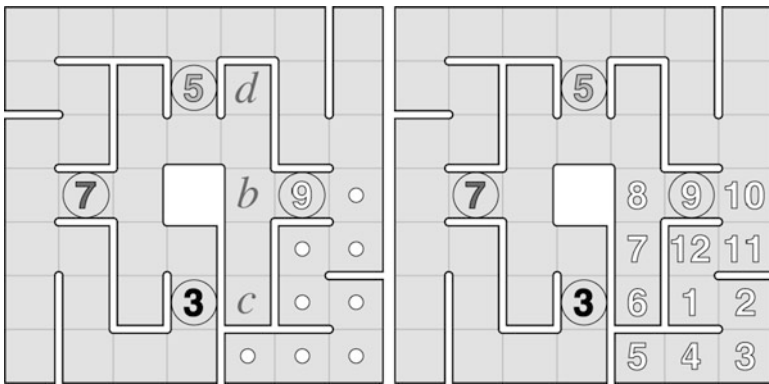


Fig. 34.5 White extends to cell *c* with known orientation

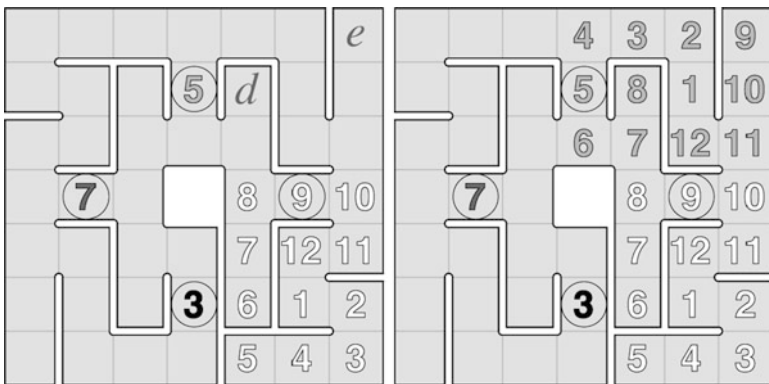


Fig. 34.6 Light grey spans cells *d* and *e* with known orientation

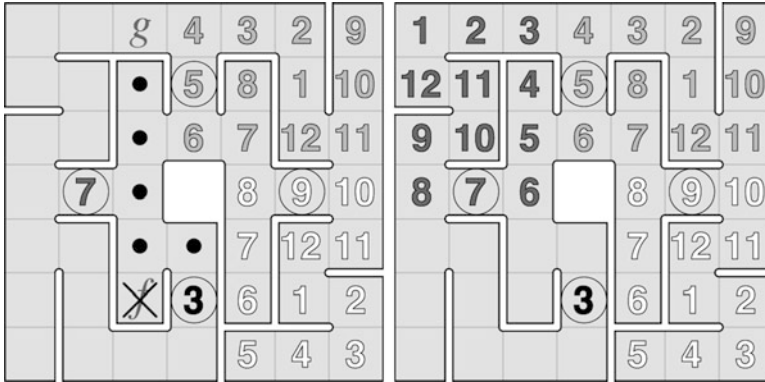


Fig. 34.7 An invalid black placement defines the *dark grey* span

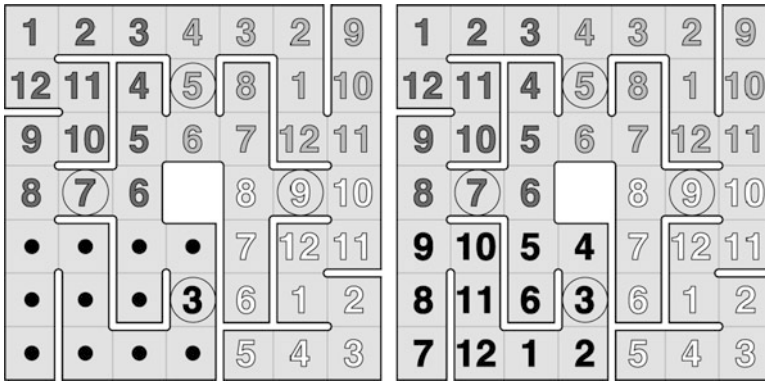


Fig. 34.8 Black fills the remaining cells with known orientation

Figure 34.7 (left) shows a tentative black extension to a nearby dead end that would isolate the cell marked *f*; black cannot make this extension so the dark grey set must visit this dead end instead. Dark grey is also the only remaining color that can visit the dead end, so the dark grey set must extend between *f* and *g* with number order determined by the light grey set (right).

All remaining cells must be black (Fig. 34.8, left) with number order determined by the dark grey set (right). This level has now been solved using pure deduction.

We emphasize again that small 7x7 levels, such as the example shown, can generally be solved easily with few strategies, whereas full size 12x12 levels can be significantly deeper and require more complex and varied strategies to solve.

Table 34.1 Number-based and color-based strategies for *Hour Maze*

Number-based strategies	Color-based strategies
1. Number count	5. Color count
2. Number availability	6. Color availability
3. Neighbor conflict	7. Color reach
4. Potential neighbor conflict	8. Color fill
	9. Color exclusion
	10. Color bounds
	11. Color isolation

34.4 Automated Solution

The first step in automated level design is to define an automated solver that can prove whether a given level is deducible or not. This section introduces a number of strategies for solving *Hour Maze* puzzles, and methods for their automation in order to perform deductive searches.

34.4.1 Strategies

The strategies employed when solving *Hour Maze* puzzles typically break down into *number-based* and *color-based* strategies. These are local in nature, and the aim of each is to successively eliminate invalid number or color choices, respectively, until only the solution remains.

Table 34.1 lists the main *Hour Maze* strategies, which were implemented for the automated solver. These are explained in further detail, with examples, in Appendix A. In the following discussion, H denotes the number of hour (color) sets. The number and/or color of a cell are *realized* if reduced to a single valid choice.

34.4.2 Deductive Search

These strategies are used to form a *deductive search* method. The algorithm works by initially setting all valid colors and numbers as potential choices for each cell (except for hint cells) then iteratively eliminating these choices until only one possible color and number remains for each cell. An attractive aspect of deductive search is that any solution found is guaranteed to be unique for that level. The basic algorithm is outlined below.

34.4.2.1 Algorithm

The search state is initialized as follows:

```
for (each cell c)
```

```

if (hintNumber[c] !=0)
{
    //Known hint
    number[c]=hintNumber[c];
    Color[c]=hintColor[c];
}
else
{
    //Unknown number and Color
    number[c]=0;
    Color[c]=0;
    numberChoices[c]=allNumbers[parity];
    ColorChoices[c]=allColors;
}

```

For each cell, if it is a hint then the known $\{number, color\}$ pair is stored for that cell, otherwise the choices for that cell are initialized to include all possible numbers and colors. `allNumbers[parity]` is a bitset that includes a flag for either all even or all odd hour values in the range 1...12, depending on the parity defined by the hint set (this step immediately halves the number of potential number choices). `allColors` is a bitset that includes a flag for all possible color values in the range 1... H .

The following steps are then iteratively applied until either a solution is found or the deductive process fails, in which case the level does not have a deducible solution. Failure will occur either when all possible choices for a cell have been eliminated or an iteration passes without a number or color choice being eliminated:

```

iterations=0;
while (!isSolved())
{
    iterations++;
    if
    (
        badCell()
        ||
        !resolveNumberChoices()
        &&
        !resolveColorChoices()
        &&
        !resolveForcedColors()
        &&
        !eliminateColorsByNumber()
    )
        return 0; //not deducible
}
return iterations;

```

`badCell()` returns true if `numberChoices[c]` or `colorChoices[c]` is 0 for any cell, in which case that cell would have no remaining valid number or

Table 34.2 Mean solution time by maze size

Maze size	Hour sets	Solution time (s)	σ (s)
5×5	2	0.0047	±0.00077
6×6	3	0.010	±0.0029
7×7	4	0.017	±0.0035
8×8	5	0.026	±0.0064
9×9	6	0.036	±0.0089
10×10	8	0.060	±0.018
11×11	10	0.089	±0.025
12×12	12	0.11	±0.028

color choices. `resolveNumberChoices()` visits each cell and eliminates any number choice that violates any of the number-based strategies described above. `resolveColorChoices()`, `resolveForcedChoices()` and `resolveColorsByNumber()` visit each cell and eliminate color choices that violate any of the color-based strategies described above (these are split into multiple methods due to their added complexity). Each of these methods return true if they result in any number or color choice elimination being performed that iteration.

`resolveNumberChoices()` also marks the number for a given cell as “realized” if it has only valid number choice remaining by setting `number[c]` to that (non-zero) value, and `resolveColorChoices()` marks the color for a given cell as “realized” if it has only valid color choice remaining by setting `Color[c]` to that (non-zero) value. `isSolved()` returns true if all cells have a realized color and number.

Note that this deductive approach is strictly strategy-based and does not involve any form of lookahead. Lookahead is not typically required due to the localized nature of the puzzle, and because several of the strategies chunk useful knowledge that might otherwise require combinatorial search.

A search result of 0 due to the complete elimination of color or number choices for a given cell, making it unrealizable, indicates that the level has no possible solution. On the other hand, a search result of 0 due to the deduction process stalling (i.e. failing to eliminate any color or number choices for a given iteration) only indicates that the level is not deducible using the given strategies. A unique solution could conceivably be found using additional strategies or some form of lookahead search for proving/disproving color and number values; we are following this point up in related work.

34.4.2.2 Performance

This deductive search approach is reasonably efficient at finding (or disproving) solutions. Table 34.2 lists the mean solution times using deductive search for maze sizes 5×5 to 12×12, and their standard deviations (σ). Timings were averaged over 3,000 randomly generated levels (ten repetitions each) using a single dedicated thread on a MacBook Pro machine with *i5* processor. Solution speeds for smaller levels are sufficient for the current application, but speed can become a consideration for full size 12×12 levels which can involve considerable combinatorial complexity (discussed shortly).

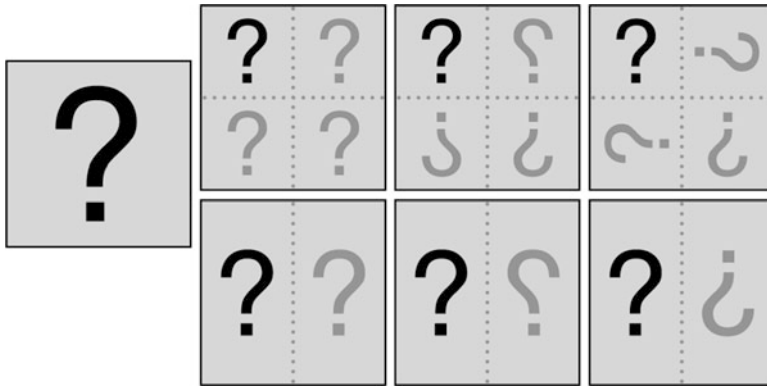


Fig. 34.9 Supported symmetries

An additional benefit of deductive search is that information about the number of deductive iterations and strategies required to solve a level is produced as a by-product of the search. This information can be harnessed to estimate the depth or difficulty of a given level, as described in the following section.

34.5 Puzzle Metrics

This section describes the metrics used to describe the most important aspects of a given level, namely its visual symmetry and solution depth (described shortly). Visual symmetry appeals to the aesthetic sense of the player and can be used to convey an impression of handcrafted rather than computer-generated design (explored further in Sect. 34.7). Deeper puzzles provide challenges that appeal to the intellect of the player, which can increase the addictiveness and replay value of a puzzle.

34.5.1 Symmetry Metrics

Symmetry metrics describe the degree of symmetry visible in each level design. There are two types of measurable symmetry: *wall symmetry* and *hint symmetry*. Seven symmetry types are supported, as shown in Fig. 34.9. These include no symmetry (left) and translation, reflection and rotation $\times 2$ and $\times 4$. Sym denotes the total number of symmetries (7).

Wall Symmetry (S_w). Wall symmetry is a measure of the degree of symmetry in a maze's wall placement:

$$S_w = \max_{[0 \leq s < Sym]} \left(\frac{agree_w(s)}{T_w} \right) \quad (34.1)$$

where $agree_w(s)$ returns the number of walls that agree with symmetry type s , i.e. the number of walls which have the full complement of translated, reflected or rotated counterparts, as appropriate for that symmetry type. T_w is the total number of walls. The final value is the maximum symmetry for any symmetry type s .

Hint Symmetry (S_h). Hint symmetry is a measure of the degree of symmetry in a level's hint placement:

$$S_h = \max_{\{0 \leq s < Sym\}} \left(\frac{agree_h(s)}{H} \right) \quad (34.2)$$

where $agree_h(s)$ returns the number of hints that agree with symmetry type s , i.e. the number of hints which have the full complement of translated, reflected or rotated counterparts, as appropriate for that symmetry type. H is the number of hour sets. The final value is the maximum symmetry for any symmetry type s .

Symmetry (S). The overall symmetry value for a level S is the average of its wall symmetry S_w and hint symmetry S_h values:

$$S = \frac{S_w + S_h}{2} \quad (34.3)$$

34.5.2 Depth Metrics

Depth metrics indicate the depth or difficulty of a given level. This may not be immediately apparent from a visual inspection of a puzzle, but can be estimated as a by-product of the deductive search process. We distinguish between *search depth* and *strategic depth*.

Search Depth (D_i). Search depth refers to the number of deductive iterations required to solve a level. If the search depth is small, this indicates that multiple deductions can be made across the level through iterations. Hence, the puzzle has more readily available clues at each point and is more easily solvable. If the search depth is large, on the other hand, this indicates that there are relatively few deductions possible per iteration, and the player must search for each opening which makes the level harder to solve. This can be represented by the following formula:

$$D_i = \max(1, \log_{10}(\text{iterations} + 1) / 2) \quad (34.4)$$

where *iterations* is the number of deductive search iterations require to solve the puzzle. The \log_{10} of this number is taken and divided by 2 then capped at 1. This gives a continuous mapping of iteration counts from 0 to 100 to the range $[0..1]$ and a value of 1 for counts above 100. Full size 12×12 levels can require over 200 deductive iterations for solution, but for the purposes of the experiment (described in Sect. 34.7) it was beneficial to focus on smaller iteration counts typical of smaller puzzles.

Strategic Depth (D_s). Strategic depth refers to the number of different strategies required to solve a level. This is estimated by disabling each strategy in turn then testing whether the level is still solvable; the final value is the ratio of strategies required to solve the level:

$$D_s = \frac{\sum_{s=0}^{Str} required(s)}{Str} \quad (34.5)$$

where Str denotes the number of available strategies and $required(s)$ indicates whether the level becomes non-deducible if strategy s is removed from the deductive search.

Depth (D). The overall depth value for a level D is the average of its search depth D_i and strategic depth D_s values:

$$D = \frac{D_i + D_s}{2} \quad (34.6)$$

34.6 Level Design

This section describes the process used to automatically generate *Hour Maze* levels. We start by describing the requirements that each level must satisfy, and some desirable properties that we would like each level to exhibit. We describe how a human designer would typically go about manually handcrafting such levels, then describe the steps we follow to automatically generate levels of specified difficulty, that attempt to incorporate at least some of these handcrafted touches.

34.6.1 Requirements

Hour Maze levels must satisfy the following mandatory requirements:

- Square grid.
- Number of cells N a multiple of 12.
- Fully connected, i.e. all cells reachable from all others by adjacent steps.
- Each cell assigned exactly one Color and exactly one number.
- Non-sequential number pairs separated by walls.
- Same-Colored cells form a single connected set.
- No 2×2 gaps of empty cells.
- Deducible solution.

2×2 gaps are forbidden as they tend to reduce the maze-like nature of the puzzle. *Hour Maze* levels should ideally satisfy the following optional requirements:

- Square footprint.
- Wall symmetry (where possible).
- Hint symmetry (where possible).

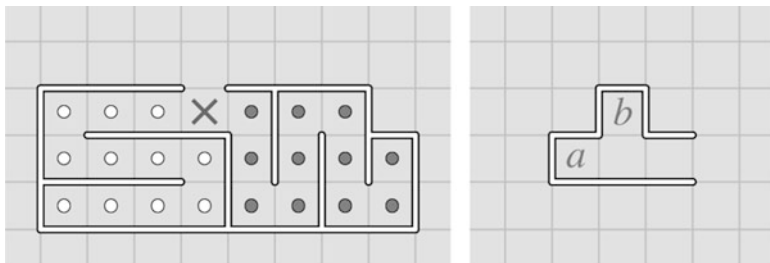


Fig. 34.10 Degenerate wall constructions

Levels with square footprints are preferred for aesthetic reasons, although this is not strictly a requirement of the puzzle. Wall symmetry and hint symmetry are mainly for aesthetic reasons, but also serve a more subtle purpose; they are more likely to give the player the impression that the level was carefully handcrafted by a human designer, which is more likely to increase their appreciation of the puzzle. The importance of this impression is discussed in more detail in Sect. 34.7.

34.6.2 Manual Level Design

Initial tests quickly revealed the difficulty of manually designing *Hour Maze* levels, especially for larger sizes. For example, Fig. 34.10 shows two degenerate wall constructions that would make level generation impossible.

The example on the left shows two corridors of 11 colored cells sharing a common exit marked X. This common cell can only be assigned one color, hence it is not possible to extend the other color to the required 12th cell to complete its set. On the other hand, the example on the right shows a corridor with two dead ends marked *a* and *b*. Any color set that visits both *a* and *b* will be bounded at both ends to give a maximum length of 3, while any color set that visits only one of these dead ends will isolate the other to give an uncolorable cell. Again, any it would not be possible to create a complete solution for any level that contained this (or a similar) wall pattern.

While it is possible to automatically test for such degenerate cases, these are only indicative of the difficulties facing the *Hour Maze* level designer. In order to avoid the need for educating the level designer in such subtleties and to allow the generation of a range of high quality content in a reasonable amount of time, it is desirable to completely automate the design process.

34.6.3 Automated Level Design

The following section describes an automated level design process that has proven to produce acceptable results. We initially tried a process that mimicked how a human designer would approach level design, that is to create an attractive wall

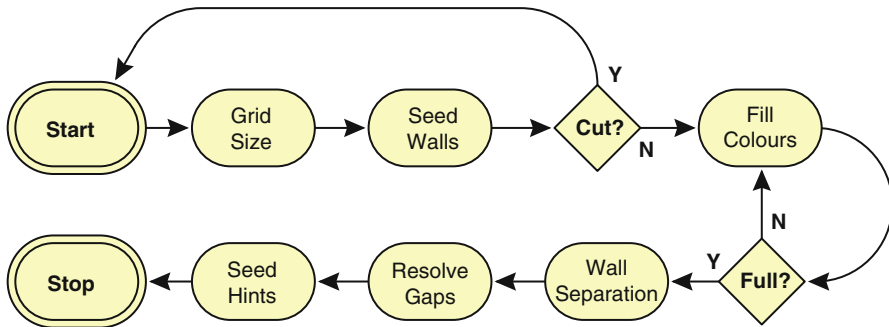


Fig. 34.11 The automated level design process

pattern first then populate it with color and number. However, this proved difficult to implement and produced poor results for smaller levels and no results at all for larger levels within acceptable times, due to problems, such as the degenerate wall patterns described above and the sheer complexity of the problem at larger sizes.

Instead, a simple paradigm shift provided a breakthrough: place color and number first then place the walls to suit. This allowed the fast generation of deducible levels (several per second) even for larger sizes, although the results tended to be rather random and without apparent purpose in design. A typical overnight run would produce 30,000+ levels, which would prove something of a headache to sort through looking for gems, which were not guaranteed. It proved beneficial to find a compromise in the automated design process and introduce symmetrical constraints at various points, which greatly improved the average quality of the resulting content.

Figure 34.11 shows a flow chart of the automated level design process. Firstly, a grid size is determined and seed walls placed until a wall set is found that does not cut the level into disconnected areas. The cells are then semi-randomly filled with color sets, with each invalid color being repeated, until all cells are colored. Walls are then placed to separate conflicting numbers, 2×2 gaps of empty cells are resolved, and initial hints are selected. These steps are now described in detail.

34.6.3.1 Grid Size

The level size is defined by the desired number of hour sets H , which must in the range of $2 \dots 12$. The design process is illustrated using a small level size of $H=4$ for clarity.

Four hour sets will require $4 \times 12 = 48$ cells, so the smallest square grid that supports this number ($7 \times 7 = 49$ cells) is chosen. The superfluous cell is flagged as unused and treated as an inverted “hole” or out-of-bounds region that is not assigned a color or number. The central cell is chosen to maximize symmetry in the design (see Fig. 34.12).

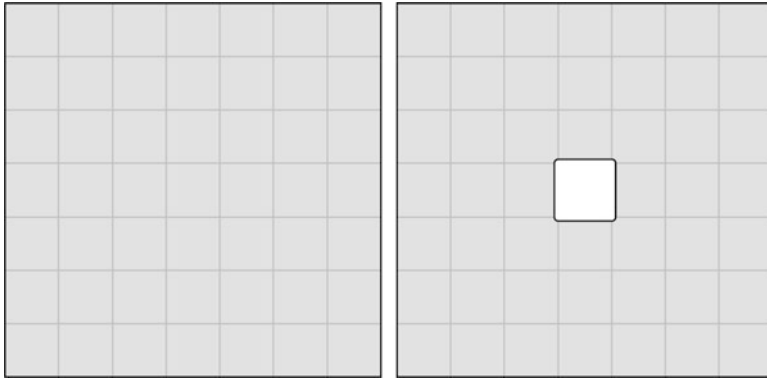


Fig. 34.12 7×7 template symmetrically reduced from 49 to 48 playable cells

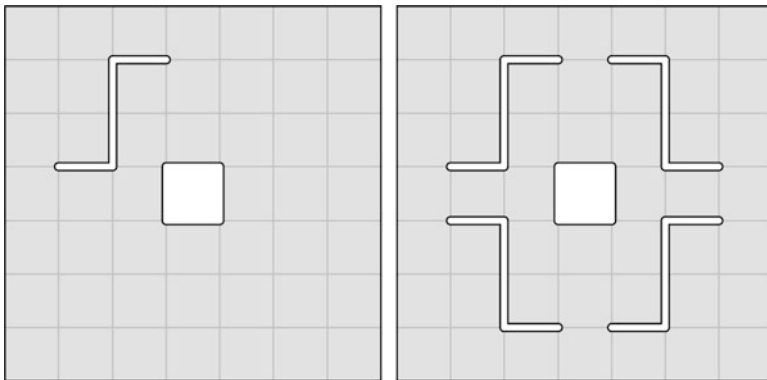


Fig. 34.13 Seed walls in *top left* quadrant, reflected horizontally and vertically

34.6.3.2 Seed Walls

To establish a symmetrical basis, a number of seed walls are placed in the grid and translated, reflected or rotated. For example, Fig. 34.13 shows four seed walls placed in the top left quadrant (left) that are reflected horizontally and vertically (right).

One of the seven symmetry types is chosen at random and the initial seed walls placed in the top left quadrant or left half as appropriate (no seed walls are placed for the “no symmetry” case). The placement of seed walls and their symmetrical repetition may result in subregions walled off from the remainder of the grid, which violates requirement three that the maze remains fully connected. Such placements are discarded and regenerated.

34.6.3.3 Random Color Walks

Once the seed walls are placed and repeated symmetrically, the *H* hour sets are added to the design one by one. The first set is added by selecting a random cell and

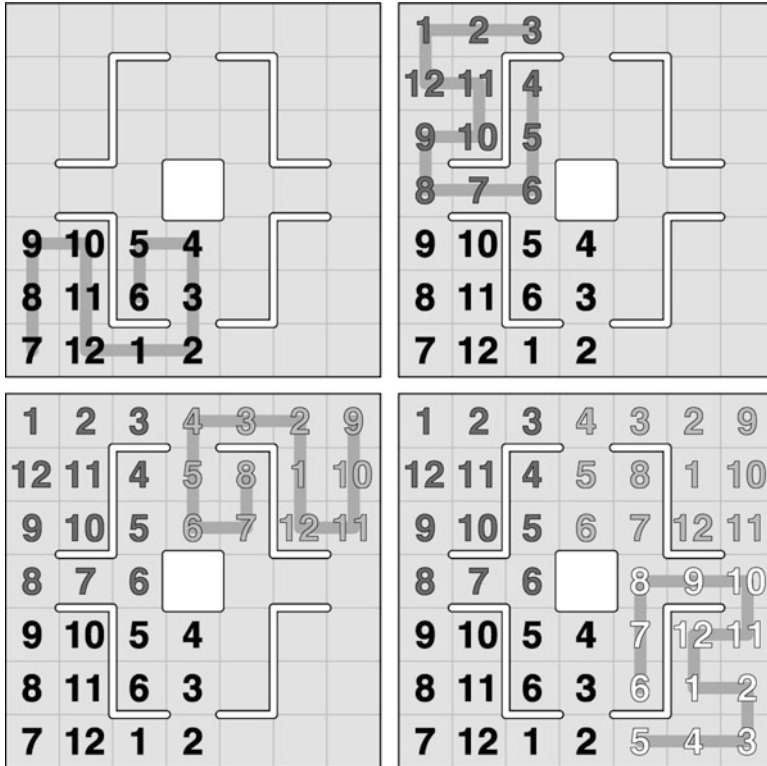


Fig. 34.14 Semi-random walk for each hour set

placing a random walk of 12 adjacent cells (Fig. 34.14, left). Subsequent hour sets are placed by extending a known cell value to an empty adjacent neighbor and completing a random walk for it, extending in all available directions until all 12 cells are assigned for that set.

The placement of hour sets may result in uncolored subregions of connected empty cells whose number is not divisible by 12. Such subregions will not allow the full placement of all remaining hour sets, so such degenerate sets are discarded and new colorings regenerated for them. Backtracking to regenerate the previously placed hour set occurs if this occurs an excessive number of times ($T=200$).

34.6.3.4 Mandatory Separation

Each cell has now been assigned a color or a number, but with no guarantee of sequentiality between hour sets or where hour sets double back on themselves. The next step is to place mandatory walls to separate adjacent non-sequential hour pairs. For example, Fig. 34.15 shows number conflicts marked (left) and walls placed to resolve these conflicts (right).

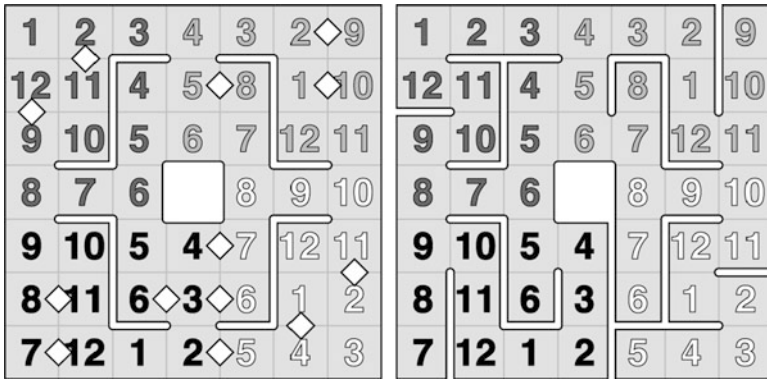


Fig. 34.15 Walls are placed between conflicting neighbors

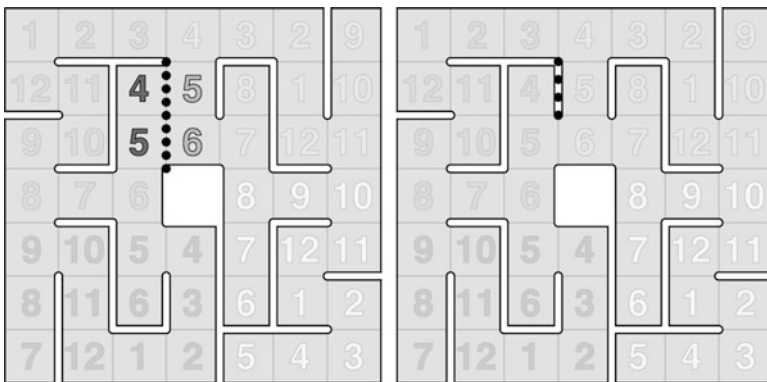


Fig. 34.16 A symmetrical wall is chosen to break the 2x2 gap

This placement of mandatory separating walls will typically reduce the symmetry of the resulting design, and is the step of the generation process most limits the impression of a handcrafted appearance. This can be seen in the example, in which the wall pattern is noticeably less symmetric following mandatory separation.

34.6.3.5 Gap Resolution

Wall placement is completed by adding walls to split any 2x2 gaps of empty cells within the design. For example, Fig. 34.16 shows the design so far with a single 2x2 gap marked, and two of the four possible walls that may split this gap dotted. The other two possible walls that would split this gap are not shown, as their placement would split a number set into two disjoint sets, hence they are illegal.

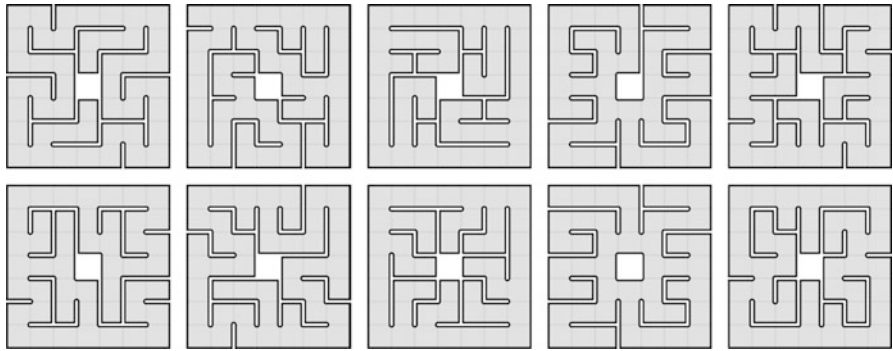


Fig. 34.17 Symmetrical maze designs

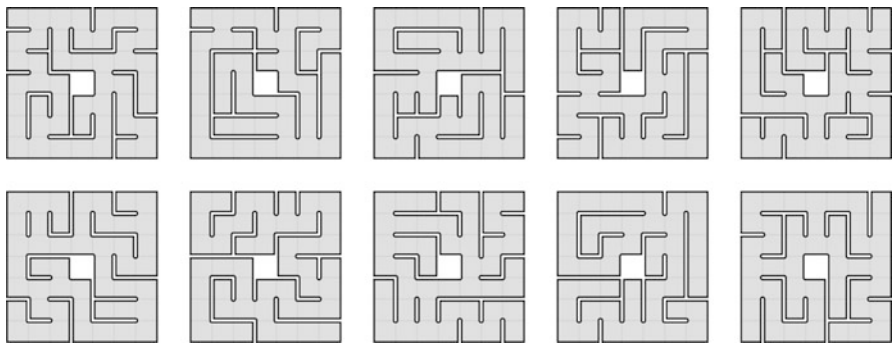


Fig. 34.18 Asymmetrical maze designs

The topmost of the two possible walls is chosen (right) as it agrees in reflective symmetry with three other walls while the other wall choice only agrees in reflective symmetry with one other wall. Gap resolution is another way in which symmetry may be encouraged in the final design.

In some cases 2×2 gaps may overlap, in which case it is sufficient to select the single wall that resolves both gaps. It is generally best to place as few walls as possible as each wall placement applies one more constraint to the level, which makes it that much easier to solve. Additional walls should only be placed if they significantly improve the symmetry or balance of the final design.

This completes the wall placement stage of the automated level design process. By way of example, Figures 34.17 and 34.18 show the difference between levels designed to encourage symmetry in the wall placement (34.17) and those that were not (34.18). We now turn to the generation of the puzzle hints, and the incorporation of further symmetry in their placement.

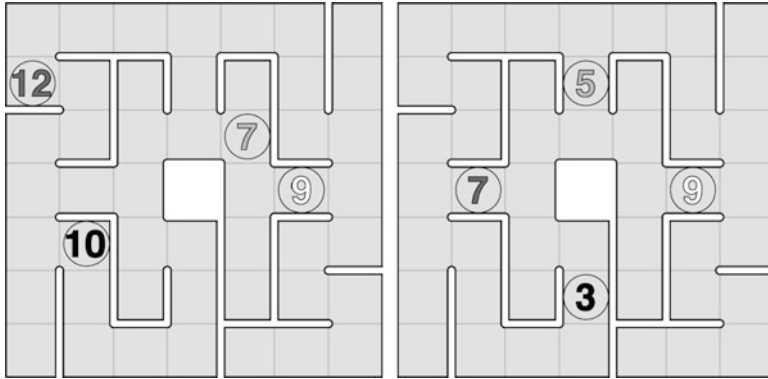


Fig. 34.19 Asymmetrical and symmetrical hint sets

34.6.3.6 Hint Selection

Having now completed the wall, color and number placements for this level, the final stage is to choose its optimal hint set. Deducibility – which has conveniently been ignored until now – becomes an important factor in this final stage, as some hint sets will allow deducible solutions and some will not.

The choice of hint set will depend on the importance that the designer places on each of the two most important design considerations: depth and symmetry. Figure 34.19 shows two deducible (hence valid) hint sets for the same level, with the set on the left favoring depth and the set on the right favoring symmetry. Rarely will a hint set favor both; these are the gems that the designer seeks.

Note that hint sets may exhibit two types of symmetry:

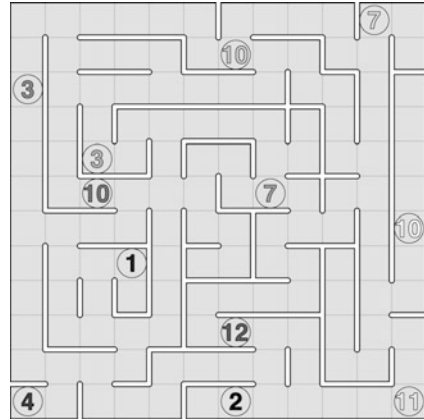
- *Geometrical symmetry*: location within the grid.
- *Algebraic symmetry*: numerical relationships between the hint values, e.g. all the same number, all different numbers (only really applicable to full size levels with the full complement of 12 hour sets), consecutive numbers, etc.

For example, Fig. 34.19 (right) shows strong geometrical symmetry and a weak algebraic symmetry in the odd sequence {3, 5, 7, 9}. The current application only measures geometric symmetry, but could benefit in future from also measuring algebraic symmetry. Random hint sets are easily generated by selecting a random number in the range 1...12 to represent the hint for each hour set. Hint sets may be conveniently stored in a single 64 bit integer. However, a more systematic approach is required if hint sets optimizing depth and/or symmetry are to be found.

The number of possible hint sets S_H for H hour sets will be:

$$S_H = 12^H \tag{34.7}$$

Fig. 34.20 Full size 12×12 level with 12 h sets



For the small 7×7 size with 4 hour sets, there will, therefore, be $12^4 = 20,736$ possible hint set combinations, a manageable number that can be tested exhaustively in less than a minute on a standard laptop machine. Of these 20,736 we can expect around 2,404 (11.6%) to be deducible on average, but only one or two (0.0075%) to be both deducible and symmetric on average.

However, full size 12×12 levels (e.g., Fig. 34.20) are another story. The search space of $12^{12} = 8,916,100,448,256$ hint set combinations could require centuries of computation to explore exhaustively, and we are currently investigating more feasible sampling approaches based on Monte Carlo methods. In the meantime, our prototype *Hour Maze* app selects hint sets for larger levels by generating a random sample over a given time period and choosing the best, which produces acceptable if not ideal results.

34.7 Discussion

Interestingly, almost 50% of the survey levels were perceived by players to be human-designed, even though they were in fact all computer-generated. This indicates the power of suggestion to affect such perceptions. More importantly, there appears to be some relationship between a player's perception of a level as being human-designed and their interest in it, although this trend was weak in these experimental results. To answer this question more definitively would require a set of puzzles handcrafted by human experts for comparison, but this will not be possible until such a comparison set exists.

Within the limited range of the 7×7 levels, players appeared to slightly enjoy those of greater strategic depth. It makes sense that strategic depth might be more important than search depth, as strategic depth is more likely to capture the way in which players would approach a level themselves (i.e. by applying strategies) rather than the iterative and exhaustive elimination of possible choices; players will find short-cuts and creative ways to chunk repeated patterns into new strategies.

The fact that hint symmetry appeared to have a slight positive effect on player interest, while wall symmetry appeared to have a slight negative effect, is interesting. This may be because walls form the substrate or background of a maze, whereas hints are more immediately a part of the solution and their placement more attributable to an authorial intelligence. The unexpected negative effect of wall symmetry may also be due to the fact that symmetric wall sets are more likely to cause strategy repetition during solution, since a strategy that works in one part of the puzzle is also likely to work in other parts of the puzzle, translated, reflected or rotated as appropriate, which is boring for the player. This effect will be emphasized for smaller maze sizes in which the limited area means that any symmetry will imply significant repetition, whereas larger maze sizes will allow both symmetry and variety to coexist in the same design.

What do these results suggest for future level designs? Firstly, that levels may benefit from being designed with as much strategic depth as possible. Secondly, that hint symmetry should probably be encouraged while wall symmetry should probably be discouraged (especially where it limits depth).

34.8 Conclusion

This chapter has introduced *Hour Maze*, a new deduction puzzle, and described automated methods for the design and solution of levels. A number of strategies for approaching the puzzle are described, including their use in a deductive search method that can quickly solve levels and yield information about the iterative and strategic depth of each level. These metrics, in addition to symmetry information about wall placement within the maze, and hint placement within the solution, provide useful information about each level.

There is a perception in similar puzzles, such as *Sudoku*, that computer-generated levels are necessarily inferior to those handcrafted by human experts. This becomes an important question for puzzles such as *Hour Maze*, for which it is possible (and in fact much easier) to automatically generate large numbers of levels relatively quickly. By maximising the number of strategies required to solve each level we can automatically affect not only the difficulty of each level, but also the potential to interest players and keep them interested, by allowing as diverse a range of solution strategies as possible. Further, by incorporating “human” touches into the level design, such as symmetric placement of the walls and hints in *Hour Maze*, we may be able to capture some of the qualities of human-designed puzzles, hence increasing the perceived value of the level designs in the player’s eyes.

I have conducted an informal user survey as a first step in this direction, in which I showed users a number of simple *Hour Maze* levels of varying difficulty and symmetry. Subjects were forewarned that the levels may be designed by either human or computer, and after completing each level were asked whether they thought it was designed by human or computer, and how interesting they found it compared to other levels they had seen. Although the responses did not produce significant results, I found it intriguing that almost 50% of levels were deemed to be human-generated

even though *all levels were in fact computer-generated*. This may simply be a symptom of subjects reverting to the null hypothesis in the absence of further information, but it seemed striking to me that such an effect could be achieved with the mere suggestion that some levels *may* be human-generated.

34.8.1 Next Steps

The Nikoli web site provides a wealth of information on the Japanese logic puzzles that inspired *Hour Maze*: <http://www.nikoli.com/en/>.

In particular, Nobuhiko Kanamoto and Maki Kaji, Nikoli's Chief Editor and President, respectively, describe what they see as the vital ingredient lacking in computer-generated puzzles:

http://www.nikoli.co.jp/en/puzzles/why_hand_made.html.

Readers wishing to see *Hour Maze* in action can find it on the iTunes App Store: <http://itunes.apple.com/us/app/hour-maze/id450067669?mt=8>

About the Author

Cameron Browne received the Ph.D. degree in computer science from the Faculty of Information Technology, QUT, Brisbane, Australia, in 2008. Currently, he is a Research Fellow at the Computational Creativity Group at Imperial College London. His research interests include Monte Carlo tree search (MCTS) methods for procedural content generation in creative domains, such as games, art and music. Dr. Browne was Canon Research Australia's Inventor of the Year for 1998, won the QUT Dean's Award for Outstanding Thesis of 2008, and was the gold medal winner at the 2012 GECCO "Humies" awards for human-competitive results in evolutionary computation, for his work on evolutionary game design.

Appendix A. *Hour Maze* Strategies

This appendix describes the key *Hour Maze* strategies implemented for the automated solver (Sect. 34.4), with examples where appropriate. These generally break down into number-based and Color-based strategies, and tend to be local in nature. Most of these strategies aim to eliminate illegal number or Color choices until the solution is deduced.

In all cases H denotes the number of hour (Color) sets. The number and/or Color of a cell are *realized* if reduced to a single valid choice. The *area* of a branch within the maze is the number of cells that it contains. A *dead end* is a cell surrounded by three walls.

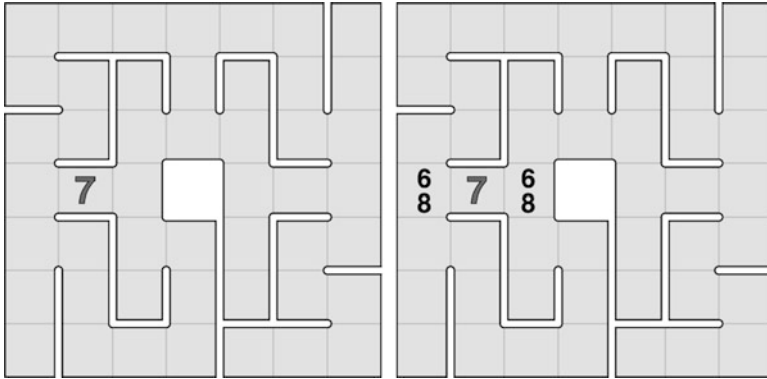


Fig. 34.21 Known numbers constrain potential neighbors

Number-Based Strategies

Number-based strategies focus on the elimination of illegal number choices.

Strategy #1: Number Count

For each hour value h , if H occurrences have already been realized then all further potential occurrences are eliminated; h has been fully accounted for.

Strategy #2: Number Availability

For each potential (or realized) Color c for the given cell, hour values h for which the pairing $\{h, c\}$ has already been used elsewhere are eliminated. There can only be one number of each Color.

Strategy #3: Neighbor Conflict

Potential neighbors of hints and other realized numbers that do not differ by $+1$ or -1 are eliminated, as adjacent neighbors must be sequential. For example, Fig. 34.21 shows a realized number 7 and its potential neighbors 6 and 8. All potential neighbors must be recalculated as each potential number choice is eliminated.

Strategy #4: Potential Neighbor Conflict

Similarly, potential neighbors that do not differ by $+1$ or -1 from at least one realized or potential neighbor are eliminated. This strategy will have a trickle-on effect across the level. For example, Fig. 34.22 shows potential numbers 6 and 8, and their

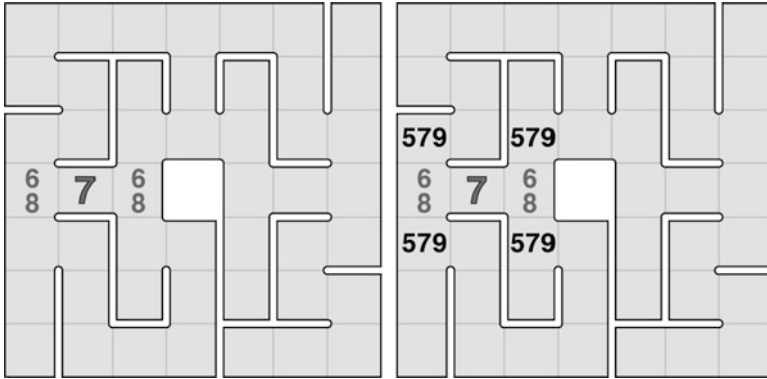


Fig. 34.22 Potential numbers also constraint potential neighbors

potential neighbors 5, 7 and 9. Again, all potential neighbors must be recalculated as each potential number choice is eliminated.

Color-Based Strategies

Color-based strategies focus on the elimination of illegal Color choices.

Strategy #5: Color Count

As soon as 12 occurrences of any Color c have been realized, remove all other potential occurrences of c . There can only be 12 occurrences of any Color; one for each hour value.

Strategy #6: Color Availability

For each potential Color c for a given cell, if all potential hour values h for that cell have previously been realized in other $\{h, s\}$ pairings, then eliminate c from that cell.

Strategy #7: Color Reach

For each Color c , eliminate c from any cell that cannot possibly be reached by that Color. If the number of cells for which Color c has been realized is R_c , then a cell will be reachable by Color c if it is not more than R_c steps away from the nearest realisation of c .

Fig. 34.23 Potential color eliminated from unreachable cells

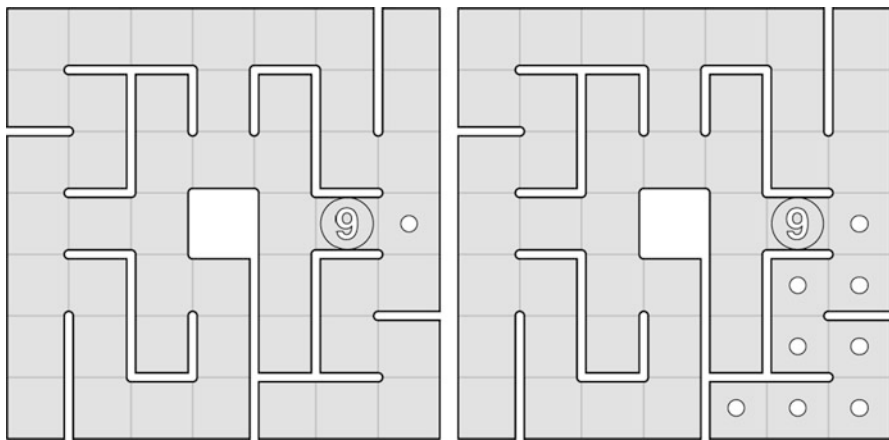
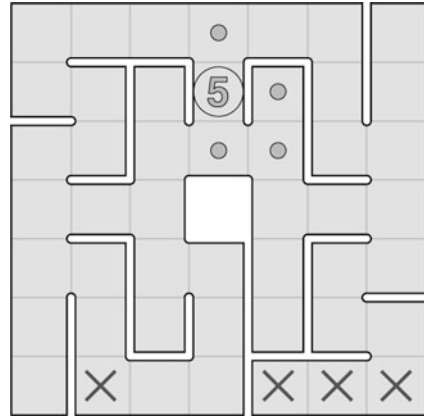


Fig. 34.24 Forced color fill of an isolated corridor

For example, Fig. 34.23 shows a level with five cells known to be light grey. The number of unrealized members of this set is $12 - 5 = 7$, so light grey is eliminated from any cell that is more than seven steps away from this set (marked X). There will always be at least one realized number of each Color due to the hints.

Strategy #8: Color Fill

For each cell adjacent to a cell with realized Color c , if that cell belongs to an isolated branch of area less than 12 then all cells along that branch must also be Color c . This is because such branches cannot possibly contain any other Color.

For example, Fig. 34.24 shows a level with two realized white cells that isolate a branch of seven cells in the lower right corner (left). No other Color can possibly reach any of these cells, hence they must be white (right).

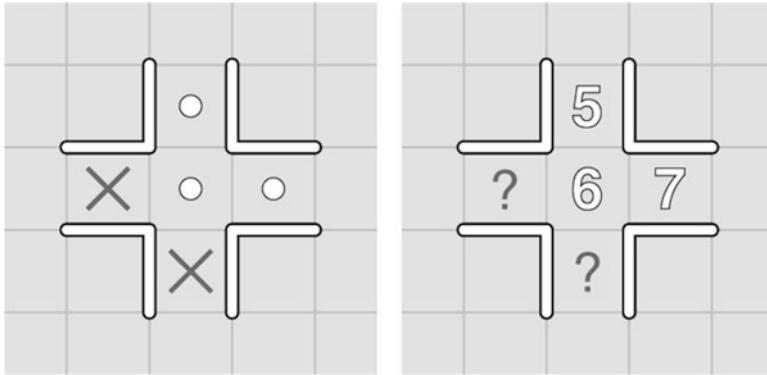


Fig. 34.25 Two neighbors of the same color exclude other neighbors

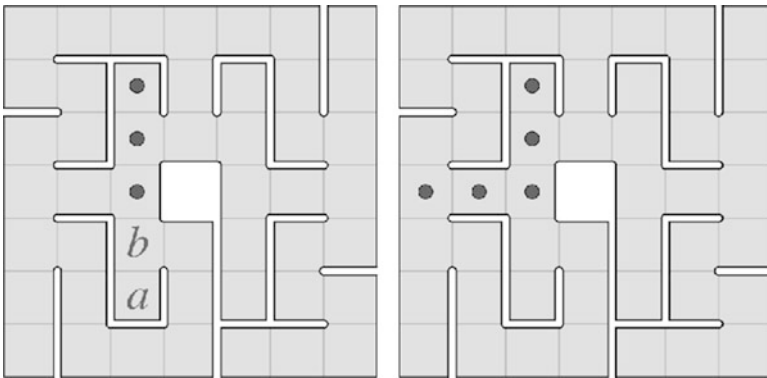


Fig. 34.26 The bounded color shown cannot enter *b*

Strategy #9: Color Exclusion

If any cell has a realized Color *c* and two of its neighbors are also realized as *c*, then *c* can be eliminated from any further neighbors. The three realized cells form a sequential path through the central cell and this path cannot branch.

Figure 34.25 shows a white cell with two white neighbors and two unknown neighbors from which white can safely be eliminated (left). Imagine that the white cells are numbered {5, 6, 7} as shown on the right. The cells marked X would have no possible number choice as 5 and 7 are already taken; paths followed by hour sets must be sequential.

Strategy #10: Color Bounds

Any dead end cell with realized color *c* bounds that Color set at one end. For example, the grey set shown in Fig. 34.26 is bounded at its topmost cell. This allows deductions to be made about possible extensions of set *c*.

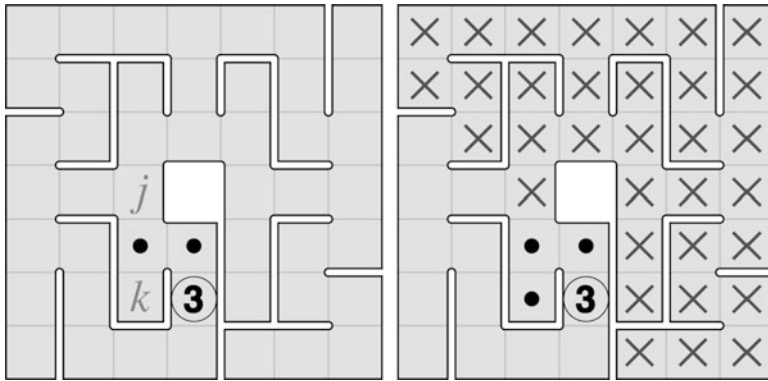


Fig. 34.27 Coloring j would isolate corridor k

For example, set c cannot extend to cell a as that would bound it at both ends for a maximum size of 5, as hour set paths must be sequential and cannot branch; grey can be eliminated from cell a .

Strategy #11: Color Isolation

Color c can be eliminated from any cell for which such a realization would isolate branches with area not divisible by 12. Such areas would not support complete hour sets and are hence invalid.

For example, cell b in Fig. 34.26 above cannot be Colored grey as that would isolate a branch of area 1 that contains cell a . Grey can safely be eliminated from cell b , hence the grey set must extend to the left as shown (right).

Figure 34.27 shows another example with three cells realized as black and possible extensions to cells j and k (left). However, extension to cell j would isolate the branch containing cell k to a single cell so black can be eliminated from cell j .

Note that the same deduction can be reached using different strategies. Cell k must be Colored black due to the Fill strategy (#8) as shown on the right, hence black can then be eliminated from cell j due to the Exclusion strategy (#9). Each level may contain such cases of strategic redundancy, which complicates the estimation of strategic depth (described in the Puzzle Metrics Sect. 34.5).

References

Ashlock, D. (2010). Automatic generation of game elements via evolution. In *IEEE Conference on Computational Intelligence and Games* (pp. 289–296). IEEE Press: Dublin, Ireland.
 Browne, C., & Maire, F. (2010). Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1), 1–16.
 Cazenave, T. (2009). Nested Monte-Carlo search. *IJCAI, 2009*, 456–461.

- Cook, S. A. (1984). Can computers routinely discover mathematical proofs? *Proceedings of the American Philosophical Society*, 128(1), 40–43.
- Crawford, C. (1984). *The art of computer game design*. Berkeley: Osborne/McGraw Hill.
- Herting, S. (2004). *A rule-based approach to the puzzle of Slither Link* (CO620 Research Project Report). University of Kent.
- Higashida, H. (2010). Machine-made puzzles and hand-made puzzles. In R. Nakatsu, et al (Eds.), *ECS 2010, IFIP AICT 333* (pp. 214–222). Springer: Berlin Heidelberg.
- Higgins, P. (2007). *Nets, puzzles, and postmen: An exploration of mathematical connections*. Oxford: Oxford University Press.
- Jing, M.-Q., Yu, C.-H., Lee, H.-L., & Chen, L.-H. (2009). Solving Japanese puzzles with logical rules and depth first search algorithm. *Machine Learning and Cybernetics*, 2009, 2962–2967.
- Kanamoto, N. (2001). *A well-made Sudoku is a pleasure to solve*. http://www.nikoli.co.jp/en/puzzles/sudoku/hand_made_sudoku.htm. Accessed July 1 2012.
- Kendall, G., Parkes, A., & Spoerer, K. (2008). A survey of NP-complete puzzles. *ICGA Journal*, 31(1), 13–34.
- Kim, S. (2008). *What is a puzzle?* <http://scottkim.com/thinkinggames/whatisapuzzle/index.html>. Accessed July 1 2012.
- Knuth, D. E. (2011). Nikoli puzzle favors. In D. E. Knuth (Ed.), *Selected papers on fun & games*. Stanford: CSLI Publications.
- La Monica, P. (2006). Much Ado about Sudoku. *CNN/Money*. <http://money.cnn.com/2005/09/19/news/newsmakers/sudoku/>. Accessed July 1 2012.
- Lee, L., Goodwin, G., & Johnson-Laird, P. N. (2008). The psychological puzzle of Sudoku. *Thinking and Reasoning*, 14(4), 342–364.
- Nikoli. (2001). Nikoli.com. <http://www.nikoli.com/>. Accessed July 1 2012.
- Ortiz-Garcia, E. G., Salcedo-Sanz, S., Leiva-Murillo, J. M., Perez-Bellido, A. M., & Portilla-Figueroa, J. A. (2007). Automated generation and visualization of picture-logic puzzles. *Computers and Graphics*, 31, 750–760.
- Pegg, E Jr. (2005). *Ed Pegg Jr.'s math games: Sudoku variations*. http://www.maa.org/editorial/mathgames/mathgames_09_05_05.html. Accessed July 1 2012.
- Pelánek, R. (2011). Difficulty rating of Sudoku puzzles by a computational model. In *Proceedings of FLAIRS 2011* (pp. 434–439). AAAI Press: Palm Beach, Florida.
- Reilly, M. (2007). *Hour Maze*. <http://hourmaze.com/>. Accessed July 1 2012.
- Rollings, A., & Morris, D. (2004). *Game architecture and design: A new edition*. Indianapolis: New Riders.
- Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. Cambridge: MIT Press.
- Sato, Y., & Inoue, H. (2010). Solving Sudoku with genetic operations that preserve building blocks. In *Proceedings of CIG'10* (pp. 23–29). IEEE Press: Dublin, Ireland.
- Schell, J. (2008). *The art of game design: A book of lenses*. Burlington: Morgan Kaufmann.
- Thompson, M. (2000). Defining the abstract. *The Games Journal*. <http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml>. Accessed July 1 2012.
- Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3), 172–186.
- Trefay, G. (2010). *Casual game design: Designing play for the gamer in all of Us*. Burlington: Morgan Kaufmann.
- Yato, T. (2003). *Complexity and completeness of finding another solution and its application to puzzles*. Masters Thesis, University of Tokyo.

ERRATUM

Magy Seif El-Nasr • Anders Drachen
Alessandro Canossa

Editors

Game Analytics

Maximizing the Value of Player Data

DOI 10.1007/978-1-4471-4769-5_35

There are some errors that should have been corrected. These are:

Frontmatter Page xiv – Eric Hazan’s affiliation should read: Ubisoft, Montreal, Canada

Page 477 – Author affiliation should read: Ubisoft, Montreal, Canada

Page 477 – Personal email address ‘eric_hazan@sympatico.ca’ should have been removed

Page 481 – The following text from paragraph 1 should have been removed: “Drachen et al. (2011) and” – this would leave the final sentence to read: “see also case study 3 in Chap. 14 for more examples.”

Page 496 – The following reference from the reference list should have been removed: Drachen, A., & Sørensen, J. R. M. (2011, June 28–July 1). Arrgghh!!! Blending quantitative and qualitative methods to detect player frustration. In Proceedings of foundation of digital games 2011 , Bordeaux, France.