An Experimental Approach to Identifying Prominent Factors in Video Game Difficulty

James Fraser, Michael Katchabaw, and Robert E. Mercer

The University of Western Ontario, London, Ontario, Canada
{jfraser,katchab,mercer}@csd.uwo.ca

Abstract. This paper explores a full factorial analysis methodology to identify game factors with practical significance on the level of difficulty of a game. To evaluate this methodology, we designed an experimental testbed game, based on the classic game Pac-Man. Our experiment decomposes the evaluation of the level of difficulty of the game into a set of response variables, such as the score. Our offline experiment simulates the behaviour of Pac-Man and the ghosts to evaluate each game factor's impact on a set of response variables. Our analysis highlights factors that significantly contribute to the game play of individual players as well as to general player strategies. This offline evaluation provides a benefit to commercial games as a useful tool for performing tasks such as game balancing, level tuning and identifying playability and usability issues.

Keywords: Dynamic Difficulty, Game Balancing, Adaptive Game System.

1 Introduction

One of the keys to the recent and continued success of the game industry has been the ability to expand to new demographics of game players outside the normal user groups [2]. A large portion of the recent success can be attributed to a developmental shift in the way players interact with gaming systems, such as Nintendo Wii, PlayStation Move and Microsoft Kinect. As the demographics of game players expands, so too will the range of players abilities and entertainment needs. Players have varying skill levels in terms of characteristics such as reaction times, hand eye coordination, and tolerance for failure. The variation in players' abilities will increase the difficulty for game designers to sort players into the usual static and preset difficulty settings of easy, medium and hard. When the game's difficulty is not correctly matched to the player's ability, the player could become bored or frustrated with the game [2], which results in reduced play time or abandonment of the game. To overcome these problems, there is a growing need to more thoroughly understand game difficulty, especially the impact of various factors and design decisions on difficulty and outcomes.

To form this understanding, experimental methods are required to collect and analyze the necessary data in a rigorous fashion. This, unfortunately, is a daunting task for any game of reasonable size and complexity. Consequently,

D. Reidsma, H. Katayose, and A. Nijholt (Eds.): ACE 2013, LNCS 8253, pp. 270–283, 2013.

the goal of our current work is to formalize an approach to studying game difficulty, in particular identifying prominent factors and determining their impact on player experience. This approach has applications to offline analysis during production to support game balancing, level tuning, and issues with playability and usability. Online applications for adjusting adaptive game systems include determining suitable factors and the granularity of the adjustment to optimize the player's experience. In this paper, we present our approach, based on a full factorial analysis methodology, and demonstrate its usefulness through applying an experimental assessment to a variant of the classic game Pac-Man.

2 Related Work

Current research on adaptive gaming systems focused on performing dynamic difficulty via three broad areas of gameplay. The first method of adjustment alters characteristics of the player's avatar. Performing adjustments to the player's character can be the most intrusive method if incorrectly performed. As the player has a high level of involvement and an increasing emotional attachment to their character, they can be sensitive to modifications and less willing to accept changes [2]. If a player identifies that an adaptive difficulty system is controlling the pace of their game play, their sense of achievement for completing difficult tasks can be diminished or extinguished [5]. If the player begins to attribute the result of their game play to external factors, they can become extremely frustrated and can completely lose interest in participation [2].

Dynamically modifying the level layout or the placement of game items or level objectives is another method of performing adaptive difficulty. Researchers have dynamically created entire levels in [11,12] or dynamically repositioned items within the world [4]. One interesting criticism is that although the adaptive system can provide that desperately needed change that comes just in time to save the player, it reduces the player's ability to formulate a consistent strategy for a level. The dynamic aspect of item value and placement reduces the player's ability to accurately assess the risk to reward ratio in accomplishing a task.

The third and most prevalent method of adaptive games is to modify the attributes and behaviours of non-playable characters (NPC). NPCs are characters outside the player's control. The advantages of adapting the NPC is that NPC's attribute information is rarely visible and players expect the opponents to increase in difficulty throughout the game. Thus, the player is less likely to notice modifications to the difficulty of the game or feel cheated by the adaptive game system when changes occur. Modifying the decision process of the NPCs provides additional variation to the game play, which can be exciting and challenging in addition to providing a range of difficulty. Research for adapting NPCs' behaviour uses a variety of approaches from machine learning for interest heuristics and balanced games [1,14], to other methods of dynamic scripting behaviours [8,9].

Current research has treated these three methods in adaptive gaming as independent in identifying difficulty and adapting the game. As areas for adaptation

are intertwined, it is important to understand their relationship to avoid unexpected emergent effects in the level of difficulty. Understanding the relationship between each of these adaptable areas, provides a greater flexibility in approaching the adaptive process. It allows the adaptive process to prioritize types of adjustments to avoid drawing attention to the adaptive system and minimize the risk in breaking the player's immersive state and level of believability in the game. The adaptive game system could avoid drawing attention to itself by making subtle adjustments to alter the NPC's behaviour during level play. Then, between levels, larger structural changes can be made to the level design that substitutes an equal level of challenge, so that the difficulty of individual NPCs can be reduced to their original values. It can also be beneficial to have multiple methods of adapting the game to add variety to the type of adjustment. Our experimental methodology allows the simultaneous evaluation of the three types of adjustments previously mentioned for adaptive gaming. It provides a consistent, rigorous method of comparing the effects of the modifications to game factors in each of these areas. Collecting data from a large number of games provides the potential to generalize how the changes impact specific strategies or particular player types.

Adaptive game research has focused on controlling a single response variable such as score or health to balance gameplay. However, a game with a balanced score or health might not produce an engaging or an immersive player state and such an adaptive system has no alternative avenues for adaptive adjustments. A single response variable might help balance the end result of the game, but it does not address the pace or progression of the game. Our research will evaluate a larger number of response variables, which provides greater expressive power in the methods of adapting the game and can provide a finer level of granularity in the selection of adjustments. By improving granularity, quantity and variety in the types of adjustments, our approach provides the adaptive system greater control over the pace and progression of the game.

3 Methodology for Game Difficulty Assessment

Our experimental methodology is designed with the objective of identifying significant game factors relating to the level of difficulty of the game. The design should be flexible and modular allowing game designers to quickly integrate this methodology in the early stages of the development process to reinforce design decisions or into the later stages as a tool for game balancing. To determine the full effect of a game factor and its significance on the response variables, we must view the effects in isolation, and in relation to other factors to discover emergent interactions. Different players will be affected by the game factors in different ways, thus we must explore each game factor's effect on a variety of player strategies. Using a variety of different player strategies increases the potential of representing a wider range of players types.

Our methodology for evaluating the significance of game factors on a set of response variables is as follows:

- Identify game factors to be modified during the experiment.
- Select a set of factor levels for each game factor.
- Identify metrics of performance goals.
- Simulate gameplay in an experimental testbed with factors at each factor level.
- Repeat the simulation to identify error rates.
- Collect the response variables related to performance goals.
- Perform the full factorial analysis, which calculates effect sizes and statistically significant terms
- Evaluate the quality of player models via the R-Sq values.
- Identify and evaluate results of significant factors on player strategies.

Our evaluation method for identifying factors with statistical significance effects on the response variables involves the use of a full 2^k factorial design. A full 2^k factorial analysis, contains k factors, each with 2 values or levels. A full factorial analysis calculates the effect sizes and statistical significance of each factor and for all possible combinations of the set of factors for each response variable. This information provides the magnitude and direction of the effect on a particular response variable. Although, this creates a large volume of terms and data to evaluate, in practice the game significance of many of these terms will be minimal and thus the amount of information stored in the adaptive game system can be greatly reduced.

4 Application of Our Approach

Pac-Man is a classic 2D game, the object of the game is to navigate through a fully visible maze and collect all of the tokens. Ghosts attempt to eat Pac-Man when in predator mode, except for a short period of time after Pac-Man has collected a power-pellet. Pac-Man is awarded points during game play for collecting tokens, power-pellets, bonus fruit, eating ghosts or completing levels. To progress past a level, Pac-Man must collect all the tokens and power-pellets in the level. At regular intervals during game play, bonus fruit items will appear on screen.

Pac-Man is a common testbed in video gaming research and is an ideal context for our research objectives because the entire game is composed of a relatively small number of game elements. However, the small number of factors will not limit the players level of engagement, as Pac-Man can provide a unique game experience due to the emergent behaviour of the opponents.

4.1 Factor Selection

In the initial stage of the experiment, we refined the list of factors to be included for definition modeling by performing a 2^k exploratory analysis. The 2^k analysis, requires a high and low value for each factor.

4.1.1 Agent Factors

Agent factors directly relate to the attributes of Pac-Man or the ghosts' behavior. Each agent has 3 active states; fleeing when the agent is escaping predators, chasing when the agent is a predator and can see the prey, and wandering when an agent is unable to view any predators or prey. Ghosts can be in a fourth state of inactivity, which occurs when they have been killed and are returning to their spawn position on the board. The selected factors and their factor levels are: the number of steps in the flee and death state time (10, 20) and Manhattan distance of the vision range (5 or 10 squares)

4.1.2 Bonus Factors

The bonus fruit will be generated at regular intervals and placed in a random empty square. The three selected bonus fruit factors and their low and high factor level respectively are: the number of steps the bonus item is available for (10,20), the perceived value of the bonus item (75, 200) and the frequency in steps at which the bonus will be generated (50, 100). By increasing or decreasing the frequency of the fruit, we alter the highest possible score that a player can achieve during a single session. Thus, we controlled for this variable by separating results into two categories for each factor level. The bonus factors provide possible insight into the perceived level of difficulty by the player, since the choice to participate in bonus tasks could indicate that the player is ready for an adjustment in the level of challenge.

4.1.3 Pac-Man SSS-AB* Algorithm

The SSS-AB* is a special case of the minimax algorithm [6]. This algorithm represents a near-optimal player capable of surviving and obtaining higher scores. Our SSS-AB* algorithm will simulate paths associated with the current intersection point to the next intersection point. This restriction improves the computation time and has limited effect on the outcome because the search depth is partially controlled by the vision factor. Pac-Man's vision range may not allow the SSS-AB* algorithm to fully simulate all the results over the set of possible successor paths, in which case the score Pac-Man can achieve given a limited view will be returned.

4.1.4 Pac-Man Weight Heuristics (PW) Algorithm

The PW algorithm sums weighted heuristics calculated using information about Pac-Man's visible world and produces a score for each possible moves; the move with the highest score is selected. For each Pac-Man heuristic, the selected weight factor levels are 0.5 or 1.0. Pac-Man's heuristics are organized into to three categories: edible goals, avoiding ghosts and global positioning. All distance based heuristics use the Manhattan distance to the objective; objectives include the collection of objects such as: tokens, power-pellets, fruits and any edible ghosts. Pac-Man must avoid ghosts while playing the game, thus a distance and direction function are used for avoiding the ghosts. The two final heuristics are: one which keeps Pac-Man away from the centroid of the ghosts, and another which

moves Pac-Man towards the centroid of the remaining items. These heuristics are modified versions of heuristics shown to be successful for improving Pac-Man's score in the work of other researchers [10,13].

4.1.5 Ghost Flocking Algorithm

The flocking algorithm is an emergent behavior algorithm, in which each member of the flock follows a set of simple rules based on the position of other visible flock members[7]. A flocking algorithm normalizes and sums the results of each of the governing rules, to decided on the best direction to move. The rules that govern our flocking algorithm are: separation, cohesion, alignment and hunger. The first three parameters place emphasis on an individual separating, moving toward and aligning direction with other members of the flock. The hunger rule places emphasis on the aggressiveness in chasing Pac-Man. For each of flocking rules, the factor levels are 0.5 or 1.0 for the weights of each rule.

4.1.6 Ghost Weight Heuristics (GW) Algorithm

The GW algorithm implements a strategy of weighted heuristics similar to the PW algorithm section. The heuristics selected are similar to those selected in the Pac-Man case; which include distance functions to objectives for protecting: tokens, power-pellets and bonus fruit. In addition, ghosts have heuristics objectives for moving towards Pac-Man, the end of Pac-Man's current path and toward and away from other ghosts. The factor levels values are 0.5 or 1.0 for the weights for each heuristic.

4.2 Performance Measures

The response variables selected for this experiment are the: 1: score (Sc), 2: number of steps (STP), 3: number of close calls (CC), 4: number of repeated steps (RSq), 5: number of fruits collected (FC), 5: number of tokens collected (TOK), 7: number of power-pellets collected (PP), 8: number of ghosts eaten (GE) and 9: the number of levels completed (LVL). Close calls occurs if a ghost comes within 2-squares of Pac-Man. Repeated step indicate Pac-Man returned to a square with no point value.

5 Conducting Experimentation

In each simulated game, Pac-Man has 3 lives and is unable to gain additional lives via points or bonus items. Due to the fact that our experiment continuously runs using the same level, if the player exceeded 350 steps, that particular life would be halted. This restriction helped normalize situations where Pac-Man was significantly better than the opponents or where Pac-Man is unable to complete the level but remains alive. Our experiment consists of four cases; one for the simulation of each of the algorithm pairs: SSS-AB* and flocking (SSS_FLOCK), SSS-AB* and GW (SSS_GW), PW and flocking (PW_FLOCK) and PW and GW algorithms (PW_GW). In our largest case, the PW_GW algorithm pair has

20 factors. The 2^k factor analysis of this case results in over a million simulations, which is then multiplied by 3 repetitions for over 3 million simulations. One of the key benefits of the full factorial analysis is that it is orthogonal, meaning the analysis can be divided and performed in a distributed method. Thus, we further separated the cases for the algorithm pairs into groups containing 10 factors, thus 1024 minimal runs. The decision to review groups of 10 factors was partially made because of the total number of factors for each algorithm, but it also minimized the intermediate file sizes and was computationally efficient. Furthermore, if cases in the analysis use the same number of factors, index information for the factorial analysis only needs to be calculated once.

6 Experimental Results

Presenting the massive volume of data collected and organized from the simulations would be verbose and would detract from the experiment goals of evaluating the application of our methodology to the domain of gaming and dynamic difficulty. Instead, we focus the discussion of our results on the main effects of selected cases from the experiments; in-depth results can be reviewed in [3]. For SSS_FLOCK, we selected the fruit frequency at a low level, as the two cases have nearly identical results. For the SSS_GW algorithm, we selected the fruit time at a high level value. For the PW_FLOCK and PW_GW algorithm, we selected the factor for avoiding the ghosts at a high level value, as it had the most positive influence on the response variables for PW algorithm.

6.1 Model Evaluation

The model evaluation indicates the percent of variation that our model was capable of explaining with the 127 highest sum of square terms. The 127 terms limitation comes from the commercial software Minitab; this evaluation method could indicate whether additional terms are required to improve the model's performance. Calculating the R-Squared (R-Sq) values provides two useful pieces of the information: it indicates the percentage of variance that our model explains and the model's accuracy in predicting other data points. This information can be used in adaptive systems to distinguish between response variables that are potentially less predictable or difficult to adapt for a particular player. The second part of the model evaluation investigates the consequences of the commercial term limitation, as not all 1024 terms could be included in the model; we calculated the lack-of-fit values for the 897 terms not included and whether their exclusion played a significant role in the results of the experiment.

The results in Table 1 identified that the most difficult and inconsistent response variables to predict and adapt for are collecting the fruit and the number of levels completed. The models for the number of close calls, steps and score showed higher levels of predictability among the response variable models and are potentially good candidates for the adjustment process. The lack-of-fit F-values ranges for the models are: SSS_FLOCK (0.16 - 0.38), PW_FLOCK (

Table 1. Model evaluation using the R-Sq (Adj) for each algorithm and all response variables

Response Variables	SSS_ FLOCK R-Sq%	PW_ FLOCK R-Sq%	SSS_GW R-Sq%	PW_GW R-Sq%
CC	83.1	70.6	63.7	69.5
FC	69.6	70.0	58.2	61.0
GE	80.4	71.0	69.3	72.8
LVL	79.5	65.1	60.4	58.1
PP	80.3	66.0	64.4	65.7
RSq	78.5	74.0	67.0	76.5
Sc	80.2	68.4	59.3	67.17
STP	79.7	74.0	67.2	77.62
TOK	80.52	67.4	65.0	68.8

0.29 - 0.39), SSS_GW (0.34 - 0.47) and PW_GW (0.33 - 0.41). The lack-of-fit testing proved the commercial limit of 127 terms did not play a significant role in limiting the results of our separated case models, although the term limit did force splitting the models into separate cases due to the large number of factors included in the design phase. The lack-of-fit results indicate that many terms were not statistically significant to the game. This highlights one of the benefits of this methodology; we identified a large number of factors game designers would have experimented with but had no impact on the results of game play.

6.2 Statistical and Game Significance

The statistically significant terms play an important role in the selection of game elements to adjust by identifying consistent terms. Statistically significant factors whose influence exceeds a minimal threshold that actually impact the player's strategy will be referred to as game significant factors. The game significant factors will have the largest effect on the response variables, as they have the greatest potential to impact the gameplay experience.

6.2.1 SSS_FLOCK Statistical and Game Significance

The results for the flocking algorithm suggest that Pac-Man is capable of avoiding the ghosts if they become overly aggressive or separated and has no difficulty in tracking and eating the ghosts. A less aggressive tactic proved effective at limiting Pac-Man's progression, by increasing the cohesion and alignment of the flock. The ghosts' vision is the most prominent statistical and game significant factor causing large decreases in all response variables, which suggests a vast improvement in the efficiency and performance of the ghosts. The flee and death state time contributes significantly to nearly all response variables, the exception is the number of fruit eaten; this supports the idea this player is not using the additional flee time to perform bonus tasks. The length of time the fruit results indicates that the player is unable to consistently improve their score via the

additional time the bonus items are available. The perceived value of the fruit not being a statistically significant main effect for more response variables could be due to the bonus item always being worth more than tokens or power-pellets. As such, it represents a goal the SSS-AB* algorithm will always attempt to complete, while minimizing uncertainty and unnecessary risks.

Table 2. SSS_FLOCK statistically significant main effects are listed with: P for positive or N for negative effect. Non-statistically significant main effects are indicated by the - symbol.

	CC	Sc	STP	TOK	GE	PP	RSq	FE	LVL
Flee State	N	P	P	P	P	P	P	-	P
Death Time	-	P	P	P	-	P	P	P	P
Fruit Time	-	-	-	-	-	-	-	P	-
Perceived Value of Fruit	-	-	N	-	-	-	-	P	-
Flock Separation	N	-	-	-	P	-	P	P	-
Flock Cohesion	N	-	-	-	N	-	N	-	-
Flock Alignment	N	-	-	-	N	-	N	-	-
Flock Hunger	P	P	P	P	-	P	P	-	P
Pac-Man Vision	P	P	-	P	P	P	P	P	P
Ghost Vision	N	N	N	N	N	N	N	N	N

6.2.2 SSS_GW Statistical and Game Significance

The GW ghosts protecting the tokens significantly decreases several of the response variables, despite the decrease in performance, there is no statistically significant decrease in the number of steps which suggests the player is capable of staying alive but is unable to complete additional points based tasks. The ghosts protecting the power-pellets or the bonus fruit item proves to be an ineffective strategy, having either no statistical significant or improving Pac-Man's performance. The remaining GW ghost factors all have a negative impact on the response variables. The results indicate that chasing Pac-Man's position does not limit Pac-Man's ability to stay alive but does minimize the ability to collect items. Chasing the end of Pac-Man's path proves slightly more effective in diminishing the number of items collected. Interestingly, while chasing Pac-Man's position increases the number of the close calls and levels completed, it reduces the score and the number of ghosts eaten. Chasing the end position of Pac-Man's current path results in a higher number of close calls while decreasing the score and the number of ghosts, fruit and power-pellets eaten. Thus, chasing Pac-Man's end position proves slightly more effective in terms of limiting a wider range of response variables. Similar, to the previous case the flee and death state time factors produce nearly all positive effects, while the ghost's vision range has produces nearly all negative effects.

6.2.3 PW_FLOCK Statistical and Game Significance

The three PW algorithm factors chasing edible ghosts, avoiding the centroid of the ghosts, and moving toward the centroid of the remaining items prove to

Table 3. SSS_GW statistically significant main effects are listed with: P for positive or N for negative effect. Non-statistically significant main effects are indicated by the - symbol..

	CC	Sc	STP	TOK	GE	PP	RSq	FE	LVL
Protect Tokens	N	N	-	N	N	N	P	-	N
Protect PP	N	P	P	P	N	N	P	-	P
Protected Fruit	-	-	-	-	-	-	-	-	-
Toward Ghosts	N	N	N	N	N	N	N	N	N
Away Ghosts	N	N	N	N	N	N	N	N	N
Pac-Man Pos	P	N	-	-	N	-	-	-	P
Pac-Man Direction	P	N	-	-	N	N	-	N	-
Flee Time	N	P	P	P	P	P	P	P	P
Death Time	P	P	P	P	N	P	P	P	P
Ghost Vision	P	N	N	N	P	N	N	N	N

have a statistically significant effect for only a few response variables. For these three cases, we observe a similar pattern in which Pac-Man successfully collects additional points from fruits and ghosts, at the expense of a shorter lifetime. So while these factors had minimal impact on the statistical significance of the response variables, these factors seemed to have altered the risk and reward ratio tempting the player to gain more points at the risk of a shorter life span. The PW algorithm takes a greater number of chances to collect the fruit. This behaviour is unlike the SSS-AB* algorithm, which uses the extra flee and death time to complete level tasks. The PW_FLOCK session results indicate that nearly all factors had a statistically significant impact on the fruit collected. The flock struggled to capture Pac-Man, especially when becoming overly aggressive, however, they were effective in stopping Pac-Man from engaging in too many bonus tasks. This suggests small modifications to the flock‘s behaviour focusing on cohesion and hunger would provide a more balanced challenge.

6.2.4 PW_GW Statistical and Game Significance

The GW algorithm factors for protecting tokens or the fruit are not statistically significant for any of the response variables. Protecting the power-pellets also proves to be an ineffective strategy, as nearly every collectable response variables has a statistically significant increase. The GW algorithm factors to pursue either Pac-Man's current position or the end of current path produce overall similar results. The two factors differed in that chasing Pac-Man's path end point is explicitly less direct and may appear less aggressive, which is supported by the decrease in the ghosts eaten. The GW algorithm proved less effective when the ghosts hunted as a cohesive unit, then when becomes more aggressive and separating to chase Pac-Man. Unlike the other cases we reviewed, increasing the vision range actually increased the number of close calls the number of ghosts eaten, which indicates Pac-Man is more effective at evading the GW ghosts and the more aggressive ghosts are frequently being eaten by Pac-Man. The GW ghosts demonstrate slightly improved performance when chasing the end point

Table 4. PW_FLOCK statistically significant main effects are listed with: P for positive or N for negative effect. Non-statistically significant main effects are indicated by the - symbol..

	CC	Sc	STP	TOK	GE	PP	RSq	FE	LVL
Edible Ghost	-	-	-	-	-	-	-	P	-
Ghost Center	-	-	-	-	P	-	-	N	-
Item Center	N	-	-	-	-	-	-	P	-
Flee Time	-	P	P	P	P	P	P	P	P
Death Time	-	P	P	P	P	P	P	P	P
Flock Hunger	-	P	P	P	P	P	P	P	P
Flock Separation	N	-	-	-	-	-	-	P	-
Flock Alignment	P	P	P	-	P	-	P	-	-
Flock Cohesion	-	-	-	-	-	-	-	P	-
Ghost Vision	N	N	N	N	N	N	N	N	N

Table 5. PW_GW statistically significant main effects are listed with: P for positive or N for negative effect. Non-statistically significant main effects are indicated by the - symbol..

	CC	Sc	STP	TOK	GE	PP	RSq	FE	LVL
Protect Tokens	-	-	-	-	-	-	-	-	-
Protect Fruit	-	-	-	-	-	-	-	-	-
Protecting PP	N	P	P	-	P	-	P	P	-
Pac-Man Position	P	-	-	P	-	P	-	N	-
Pac-Man Direction	P	-	-	P	N	P	-	N	-
Ghosts Toward	-	-	-	-	-	P	-	-	-
Ghosts Away	N	N	N	N	N	N	N	-	N
Flee Time	N	P	P	P	P	-	P	P	-
Death Time	P	P	P	P	-	P	P	P	P
Ghost Vision	P	N	N	N	P	N	N	N	N

of Pac-Man's path. Interestingly, chasing the end point of Pac-Man's path produced less direct challenge but ultimately produced comparable overall results to chasing Pac-Man's current position. The overall result of these cases is nearly identical for most response variables, but gameplay would provide less conflict and likely a less stressful game session to the players.

6.3 Limitations

One of the largest caveats to examining game difficulty using a full factorial analysis methodology is the exponential growth of calculations that occurs for the inclusion of each additional factor. Fortunately, one of the strengths of the full factorial analysis is that it is orthogonal, which allows the calculations to be separated and calculated independently, reducing some of the strain of the data size issues. The complexity of the full factorial statistical analysis scales well

and remains quite simple in theory, in practical non-distributed calculations exceeding 7-8 factors cause the memory and computational requirements to exceed even the capabilities of current statistical software programs, such as SPSS, R and Minitab and evaluation might best be accomplished using a "Big Data" approach. Our experiment illustrates that even in the analysis of a smaller video games such as Pac-Man, careful consideration must be used in the methods of calculating results and managing the volume of data.

Within our experiment, the player's behaviour and development remained constant, meaning that the goals or strategies of the player never changed. Within this context, it was easier to develop player models and accurately identify statistically and game significant factors to the response variables. However, real player models are adaptive and progress as players become more experienced and begin to explore new strategies and techniques. Thus, a full player model would require progression from one player state to another; as such, it is more accurate to view the models in our results as snapshots of factors which affect a player's performance and not as complete player models. A player's type will be more accurately described via a collection of models, as they progress throughout the game, and the current player model will continually need to be re-evaluated and set to the closest matching player model. Identifying, which player model accurately describes the player's current state is not addressed nor required in our experiment but is a complex task which requires consideration in an adaptive gaming system.

7 Conclusions and Future Work

Our research investigated a methodology to identify game factors which altered the simulated player's performance on a set of response variables and the difficulty of the game. Understanding the relationship between game factors and their impact on level of difficulty and thus the player's performance is the first step toward customizing gameplay to improve the player's emotional investment and experience. The methodology used in this experiment examines factors from the three main types of adaptive adjustments relating to the player's avatar, the level layout and item placement and the behaviour of non-playable characters. Our research demonstrates the ability to quantify each factor's impact on the player's performance on a set of response variables. A small consistent set of factors played prominent roles in altering the performance of the player for all response variables. Although, intuitively, a number of these factors could be predicted as being prominent factors, it is important to identify and quantify impact, granularity and predictability.

The results of our analysis provided interesting insight into the interaction of game factors and algorithms which produced emergent behavior. This type of information can be difficult to discover during testing and can be valuable for game designers in understanding unintended changes in difficulty. The analysis highlighted interesting properties relating to the perceived level of challenge; when the player engages in bonus tasks, it often resulted in a diminished overall

performance rather than additional points scored by the player. Given this result, game designers could strategically place bonus items or tasks in areas with lower levels of interest or challenge. The placement of bonus items could be gradually adjusted into more challenging areas, which could be used to gauge the level of risk the player is comfortable competing against.

Commercially, this methodology has the potential to be effective for testing new features and tuning final factor settings for release. An advantage of this methodology is that the factorial analysis is completely independent of the factor level values or game values selected, so integration of this subsystem requires only identifying response variables to collect information about and thus this subsystem can easily be injected into a wide variety of games. It can be easily integrated into commercial products during the testing phase and due to the independence and modularity of these methodologies as a sub-system, it can be easily removed before the product is released. The ability of the analysis to evaluate any modifications to the player attributes, NPCs or level factors offers the advantage of a wider variety of adaptive decisions.

7.1 Future Work

Continuing with the design goals of this research, future research will focus the structure of our adaptive system on being a viable to the progression of both commercial and academic game development. Our future work will utilize the results of statistical methodology presented in this paper, in an adaptive game prototype. The adaptive prototype will adjust game factors during gameplay in an attempt to control the overall outcome on a selection of response variables representing difficulty. The goal of future research will be to successfully adjust game factors for multiple player sessions and to control the results for a single or multiple response variables. Our adaptive game prototype will further investigate the goals of an adaptive system and issues relating to game progression and conflicting difficulty adjustment goals.

References

1. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Challenge-sensitive action selection: An application to game balancing. Intelligent Agent Technology, 194–200 (2005)
2. Bateman, C., Boon, R.: 21st Century Game Design. Charles River Media (2006)
3. Fraser, I.J.: Game challenge: A factorial analysis approach. Master's thesis, University Of Western Ontario (2012), http://ir.lib.uwo.ca/etd/563/
4. Hunicke, R., Chapman, V.: Ai for dynamic difficulty adjustment in games. In: Challenges in Game Artificial Intelligence AAAI Workshop, pp. 91–96 (2004)
5. Miller, S.: Auto-Dynamic Difficulty. Published in Scott Miller Game Matters Blog (2004), http://dukenukem.typepad.com/game-matters/2004/01/autoadjusting.html
6. Plaat, A., Schaeffer, J., Pijls, W., Bruin, A.: SSS*= AB+ TT (1995)
7. Reynolds, C.: Flocks, Herds and Schools: A Distributed Behavioral Model. ACM SIGGRAPH Computer Graphics, 25–34 (1987)

8. Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E.: Difficulty scaling of game ai. Intelligent Games and Simulation, 33–37 (2004)
9. Spronck, P.H.M.: Adaptive game ai. Ph.D. thesis (2005)
10. Szita, I., Lorincz, A.: Learning to Play Using Low-Complexity Rule-Based Policies: Illustrations through Ms. Pac-Man. Artificial Intelligence Research pp. 659–684 (2007)
11. Togelius, J., De Nardi, R., Lucas, S.: Towards automatic personalised content creation for racing games. Computational Intelligence and Games, 252–259 (2007)
12. Togelius, J., Lucas, S.: Evolving controllers for simulated car racing. Evolutionary Computation, 1906–1913 (2005)
13. Yannakakis, G., Hallam, J.: Evolving opponents for interesting interactive computer games. In: From Animals to Animats, pp. 499–508 (2004)
14. Yannakakis, G., Hallam, J.: A Generic Approach for Generating Interesting Interactive Pac-Man Opponents. In: IEEE Symposium on Computational Intelligence and Games, pp. 94–101 (2005)