Chapter 2
An Overview of Serious Game Engines and Frameworks

Brent Cowan and Bill Kapralos

Abstract Despite the growing popularity and widespread use of serious games, the development of effective serious games is difficult, requiring an appropriate balance between game design and instructional design. Although there are fundamental differences between games developed purely for entertainment compared to those developed for "serious" purposes, there are currently no standard development tools specifically intended for serious game design and development available that encourage developers to follow a set of best practices. Rather, developers of serious games often rely on existing game engines and frameworks that are specific to entertainment-based game development. Given the availability of a large number of game engines and frameworks, deciding on which one to use to develop a serious game may be difficult, yet the choice of engine or framework can play a significant role in the development process. In this paper we present the results of a literature review that examined the frameworks and game engines that are used to develop serious games. We provide a list of the most commonly used frameworks and game engines and summarize their features. Knowledge of the frameworks and game engines that are most popular and details regarding why they are popular may prove to be useful to serious games developers seeking such tools. The chapter ends with a brief discussion regarding a framework that is currently being developed specifically for the development of serious games. Through consultation with the potential users of the framework (serious games developers), the framework aims to strike a balance between ease of use and functionality, while providing the user with the necessary options and tools to ideally develop effective serious games.

Keywords Serious gaming · Virtual simulation · Game engine · Game development · Framework · Review

B. Cowan
Faculty of Science, University of Ontario Institute of Technology, Oshawa, ON, Canada

B. Kapralos (✉)
Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON, Canada
e-mail: bill.kapralos@uoit.ca

A.L. Brooks et al. (eds.), *Recent Advances in Technologies for Inclusive Well-Being*, Intelligent Systems Reference Library 119,
DOI 10.1007/978-3-319-49879-9_2

2.1 Introduction

The idea of using games for purposes other than entertainment was first formulated in a book titled 'Serious Games' by Clark C. Abt in 1975. Abt introduces the concept of serious games and defines them by stating: "We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement" [1]. The examples discussed by Abt were limited to table-top ("board") games as video games were still in their infancy in 1975. In 2002, motivated by Clark Abt's book 'Serious Games', David Rejeski from the Woodrow Wilson Center for Scholars added the term serious games to a report Ben Sawyer prepared titled "Improving Public Policy through Game-based Learning and Simulation" [47]. The expression "serious game" may be seen as a contradiction, or a tautology [7]. More specifically, if games are fun, how can they also be serious? It could even be argued that games have an evolutionary background as instruments for survival training [7]. Although no particularly clear definition of the term is currently available, "serious games" usually refer to games that are used for training, advertising, simulation, or education, and are designed to run on personal computers or video game consoles [55]. Serious games have also been referred to as "games that do not have entertainment, enjoyment, or fun as their primary purpose" [34]. A serious game can more formally be defined as an interactive computer application, with or without a significant hardware component that (i) has a challenging goal, (ii) is fun to play and/or engaging, (iii) incorporates some concept of scoring, and (iv) imparts to the user a skill, knowledge, or attitude that can be applied to the real world [5]. The terms "serious game" and "educational game" are often used interchangeably. However, the primary purpose of a serious game is not necessarily educational. For example, the America's Army series of games serve as a recruitment tool designed to persuade young people to consider a career in the U. S. military [59]. Educational games are typically viewed as a subset of serious games, and are mainly developed for use within kindergarten to grade twelve (K-12) education [42]. Education generally refers to the acquisition of knowledge, while training refers to the acquisition of both skills and knowledge. Educational games generally focus on the acquisition of knowledge while using entertainment as a motivator.

Serious games "leverage the power of computer games to captivate and engage players for a specific purpose such as to develop new knowledge or skills" [13] and greater engagement within an educational setting has been linked to higher academic achievement [50]. In addition to promoting learning via interaction, there are various other benefits to serious games. More specifically, they allow users to experience situations that are difficult (even impossible) to achieve in reality due to a number of factors including cost, time, safety, and ethical concerns [54]. Serious games may also support the development of various skills including analytical and spatial, strategic, recollection, and psychomotor skills, as well as visual selective attention [36]. Further benefits of serious games may include improved

self-monitoring, problem recognition and solving, improved short- and long-term memory, and increased social skills [36]. Serious games are being applied within a wide range of applications including, medical/surgical skills development [3], military strategy training [43], and interpersonal skills development [8]. Given the ubiquity of video game use across a large demographic (i.e., males, females, youth, and the elderly), and their ability to engage and motivate learners, the popularity of serious games has seen a recent surge across many areas of education and training.

Despite the growing popularity of serious games and the benefits they afford, the development of effective serious games is a difficult and time consuming process, requiring an appropriate balance between game design and instructional design. It has been suggested that a lack of proper instructional design will lead to ineffective serious games or "instructional games" [4, 26]. Further complicating matters, there are currently no standard development tools available that emphasize, and encourage developers of serious games to follow best practices. Many serious game developers are using tools that were developed for the creation of commercial entertainment games instead of tools designed specifically for education and training. Serious games are often developed by developers with limited knowledge regarding a game's educational content and possess limited, if any knowledge of instructional design.

In order to understand how serious games are currently being developed, we begin by examining the development tools used to create them. Here, we present the results of two literature reviews that were conducted across three databases. The first survey compiled a list of development tools that were frequently mentioned in academic writing. A search was performed for each framework using three separate search terms. By measuring the number of search results for each framework in relation to each search term, we determined which development tool or framework is the "most discussed academically". In addition to measuring the notoriety of each framework/engine, separating the results by search term allows us to examine the context in which the framework/engine was frequently discussed. The second survey was conducted to determine which tools were utilized most often in the creation of a serious game. By knowing which tools are popular among the developers of serious games, we are able to investigate the features that are most important to developers. The toolset utilized to create a game may also provide insight regarding the size of the team, their skill level, and the project's budget.

2.1.1 Paper Organization

The remainder of this paper is organized as follows. Section 2.2 provides background information, and more specifically, an overview of existing frameworks and game engines. The results of a literature review that examined the tools (game engines and frameworks), used to develop serious games are also presented. This review led to a compiled list of the frameworks and game engines that are the most popular among serious games developers. A discussion of the survey results and

concluding remarks are provided in Sect. 2.3. This includes a summary of the most popular game engines and frameworks along with common features that are important to developers of serious games in addition to a description of our ongoing work that is seeing the development of a serious game framework for medical education. This framework is intended to bridge the gap between the specific needs of serious game developers and the development tools they currently have available to them.

2.2 Game Engines and Frameworks

With respect to entertainment-based video games, early video games such as Pong were designed to run on hardware that was not well suited for video game development. With little in the way of processing power or memory to work with, games were typically written completely from scratch in assembly language (a low-level language). The close link between the game "code" and the hardware prevented the code from being re-used [6]. As hardware capabilities improved, higher level languages such as C/C++ and Java gradually replaced assembly language for game programming, leading to greater code re-use. Over time, many game companies accumulated a library of well-tested reusable code. To further reduce production time, simplified Application Programming Interfaces (APIs) and external tools such as level editors were developed. id software began licensing their Quake engine to other companies as an additional source of revenue in 1996 [31]. Other companies such as Epic Games soon began licensing their game engines (Unreal) too. There are now hundreds of commercial game engines competing for licensees. Modern game engines provide advanced rendering technologies, simple tools for content creation, allow game developers to reuse code, and decrease development time and costs [31]. Sherrod [49] defines a game engine as "a framework comprised of a collection of different tools, utilities, and interfaces that hide the low-level details of the various tasks that make up a video game". The terms "game engine" and "framework" are often used interchangeably. For the purpose of this paper, the term game engine refers to the functionality and features that become part of the completed game. A framework includes a game engine in addition to tools and interfaces that simplify the process of game development (this is depicted graphically in Fig. 2.1).

Listed below are the main features provided by most modern game engines. The code responsible for providing this functionality becomes a part of the finished game.

- **Scripting**: Simple code that can be written to control game objects and events.
- **Rendering**: The speed and accuracy by which a three-dimensional (3D) scene is generated, as well as the visual effects provided.
- **Animation**: The movement and deformation of objects, such as characters.

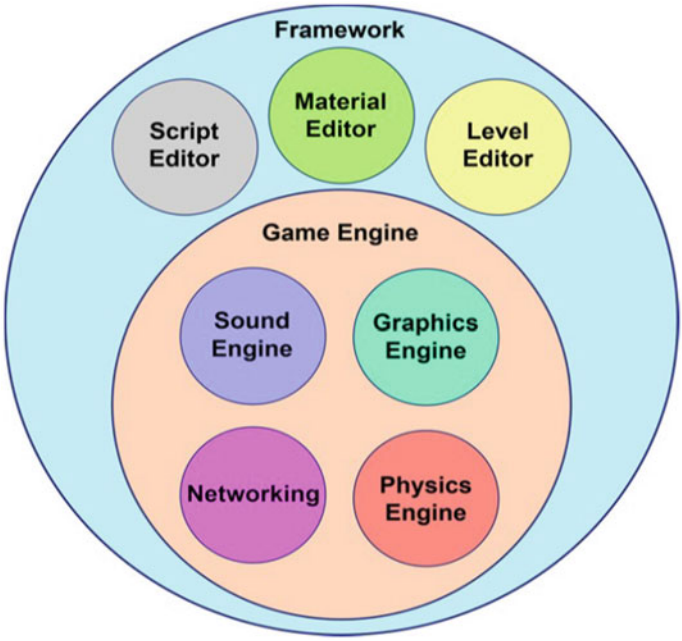

Fig. 2.1 The relationship between game framework, game engine, and their constituent components

- **Artificial Intelligence**: Steering behaviors, such as pursuing, dodging, and fleeing are combined with path finding.
- **Physics**: Objects respond accurately to applied forces or pressures (e.g., when colliding).
- **Audio**: Spatial rendering of audio allows sounds to have a location within the environment. Digital Signal Processing (DSP) is used to add variation as well as environmental cues such as reverberation.
- **Networking**: Allows players to interact with other players within the game by sharing data through a network.

Game creation frameworks generally provide a Graphical User Interface (GUI) which often ties together several editors. Listed below are some of the editing tools commonly included with game creation frameworks:

- **Level Editor**: Also known as a "world editor", this tool aids in the creation of two-dimensional (2D) or three-dimensional (3D) environments (game levels, maps, etc.).
- **Script Editor**: Scripts can be attached to objects selected in the level editor to customize their behavior.

- **Material Editor**: Shader (programs that provide a procedure for rendering surface details), code is edited and combined with images to form the surface of objects or to create visual effects.
- **Sound Editor**: Sound level, attenuation and other settings can be combined with filter effects provided by the sound engine.

Level and script editors may also be part of a game engine, and, at times, these components may be purposely included with a finished (shipped) game to encourage modification ("modding") of the game itself by the users. Game modding refers to additional game content such as new missions, items, and environments created by fans of the game, not by the developer.

2.2.1 Game Engines for Serious Games

Developers of serious games often use existing (entertainment-based) game engines and frameworks to develop serious games despite the fact that they have different needs than developers of "pure entertainment" games [41]. More specifically, serious games are often produced by small teams with limited budgets when compared to companies specializing in entertainment-based games whose budget often times is millions of dollars.

Although the development of games for entertainment often requires considerable planning, there are fewer design restrictions when compared to the development of serious games. Entertainment games must be fun and engaging; the game's content may be designed around the chosen gameplay mechanic and the designers are free to modify the design throughout the development process. Developers of serious games often begin with content that exists outside of the game world and, therefore, this content cannot be changed [4, 26]. For example, a serious game for surgical skills training must strictly adhere to the process and the equipment used in the surgical procedure in order for the skills acquired in the game to translate into real world application. Being games, serious games should also be fun and engaging, though this may not necessarily be their primary purpose.

A serious game development team may not include team members skilled in every area of game development. For example, an engine capable of rendering very large, highly detailed environments might not appeal to a team that lacks the art and modeling skills necessary to construct such an environment. In addition, some frameworks and engines are specialized for building a specific type of game, such as a first person shooter (FPS). Selecting the right engine is dependent on the type of game being developed and the skill set of the development team.

2.2.1.1 Engine Survey 1: Engines/Frameworks Most Discussed in Academic Literature

Although having knowledge of the frameworks and game engines that are popular amongst the developers of serious games will allow for the features most important to developers to be easily examined, very little prior work has considered this. As a result, two literature surveys to determine the frameworks and engines most commonly used by serious games developers (i.e., the most popular frameworks and engines) were recently conducted. The first survey (Survey 1) involved searching the following three data bases for game engines and frameworks that are most popular among serious games development: (i) Google Scholar, (ii) the ACM Digital Library, and (iii) the IEEE Xplore Digital Library. The Google Scholar search was conducted over a period of three days (August 2–5, 2015), while the ACM and IEEE Digital Library searches were conducted over a period of four days (August 3–7, 2015). The search terms "serious game", "educational game", and "simulator" were used to reveal more than 200 academic publications related to serious games and game development. Each of these publications were then scanned for the names of game engines without regard for the context in which the engine was discussed. By counting the number of publications that mention a framework by name, each framework was given a score signifying its notoriety among academic publications. The top 20 most mentioned game engines and frameworks were deemed worthy of further investigation (this initial investigation was conducted to limit any potential personal bias).

For each of the top 20 game engines, Google Scholar, the IEEE Xplore Digital Library, and the ACM Digital Library were used to survey the engine's popularity based on the number of search results returned. In each case, the engine name was combined with the search terms described above ("educational game", "serious game", or "simulator"). Many game engines, including Unity, Unreal, Torque, Half-Life, and Flash, have names that are common words in the English language and whose meaning may not necessarily have any relation to game engines or frameworks. For example, the term "half-life" is a commonly used in the physics domain to describe the decay rate of radioactive elements, but has no gaming reference. "Half-life" could also refer to a game by the same name, and not the game engine itself. To separate the search results specific to game engines and frameworks from those that did not relate to game development, the first 30 documents for each search term were manually reviewed. The approximate percentage of search results relating to the game engine was determined by dividing the number of documents found that referred to the engine by the total number of documents reviewed. The results were then normalized to ensure that each of the three search terms were equally weighted. For example, the search term "simulator" returned many more search results than the term "serious game".

The results of Survey 1 are summarized in Fig. 2.2 where the *x-axis* is a list of the top ten most discussed (academically) engines and frameworks, and the *y-axis* represents the normalized occurrence (the number of search results) for each of the three search terms ("educational game", "serious game", and "simulator").

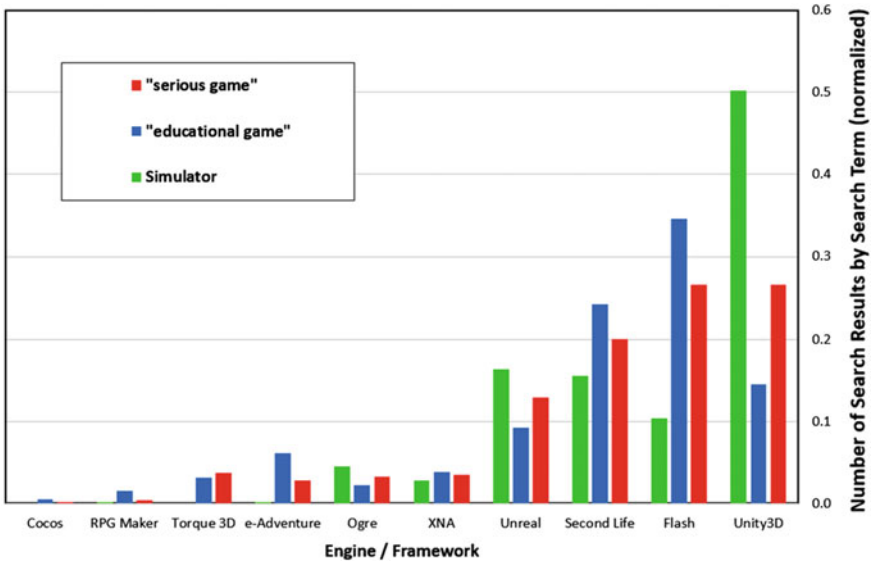

Fig. 2.2 Game engine/framework versus normalized occurrence (number of search results) by search term

The results of this survey provide an indication of the game engines/frameworks that are mentioned most frequently in academic writing. The results do not take into account whether or not the authors view the framework/engine favorably, and do not directly relate to the number of games made with each engine. For example, many of the publications that mention Second Life do not use it as a framework for their own content creation. Instead, Second Life is frequently discussed in relation to the psychological or social impact of virtual worlds. Adobe Flash is often discussed in papers relating to educational games. Unity and Unreal are frameworks capable of rendering highly realistic 3D graphics, so it is not surprising that they were frequently mentioned in papers relating to simulators.

2.2.1.2 Survey 2: Engines/Frameworks Most Utilized by Game Developers

After the completion of Survey 1, Survey 2 was conducted (August 7–11, 2015) to determine the engines or frameworks that serious game developers are using to create their serious games. This survey consisted of conducting a search within Google Scholar, IEEE Xplore Digital Library, and ACM Digital Library, using the search term “serious game”. Five hundred peer reviewed papers were downloaded from the three databases. These papers were manually reviewed to determine which

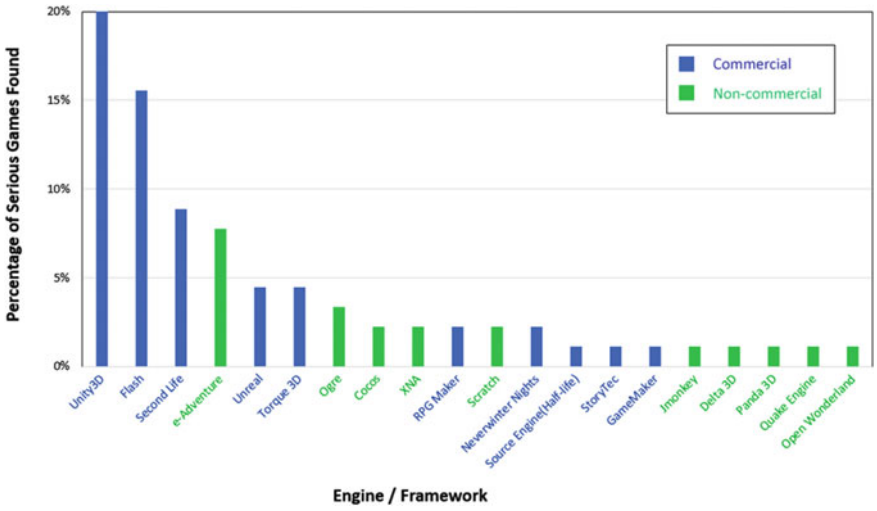

Fig. 2.3 Game engine/framework versus the percentage of serious games that were created with it

game engine the authors used to create a serious game. The majority of the papers that were downloaded (approximately 82 %) were disqualified since there was no mention regarding the framework/engine used, described more than one serious game, or described a serious game that was developed from "scratch" without the use of a specific framework/engine. Papers were also removed from the list if they featured a game that had been discussed in a previous paper, or if the first author matched an author of a previous paper. Only 90 out of 500 papers satisfied the above requirements and were used to determine the most popular engines utilized by serious game developers. The results of Survey 2 are summarized in Fig. 2.3 where, for each framework/engine, the percentage of games developed with it is provided. The frameworks/engines shown in green are freely available, while the frameworks/engines shown in blue are commercial products (although many of them are free for non-commercial use).

2.2.2 Notable Frameworks and Engines

A detailed description of the top ten most utilized frameworks/engines (as revealed by Survey 2) is provided below. The descriptions are not intended to be a review or evaluation the frameworks/engines, but rather an objective look at the main features that each framework/engine has to offer.

2.2.2.1 Unity

The Unity game creation framework was developed by Denmark-based Unity Technologies. Unity combines a powerful rendering engine that incorporates the nVidia PhysX physics engine [15]. Many successful commercial games have been developed using Unity, including various AAA titles such as Call of Duty Strike Team, and Rain. A "What You See Is What You Get" (WYSIWYG) level editor is combined with an intuitive terrain sculpting tool allowing developers to quickly generate realistic 3D environments. Objects in the environment can be visually selected, and changes can be made to their variables and scripts. Users can press the "Play" button at any time to preview the running game and test their changes [23]. Unity is non-game-type specific, allowing for 2D or 3D rendering [58].

Efficient multiplatform publishing at an affordable price has made Unity a favorite for independent developers and non-commercial projects. After a game has been created once in Unity, it can be exported to a wide variety of platforms, including all major game consoles and many mobile device platforms including Android and iOS, with only minor changes (such as reducing the file size or complexity of 3D assets) [58]. Figure 2.4 illustrates a sample screenshot taken from "Sort 'Em Up", a serious game that was commissioned by the Regional Municipality of Durham (Ontario, Canada) to allow residents to test their knowledge about recycling in a first-person shooter manner.

Access to the Unity engine source code is not provided with any of the standard licenses. All game logic is written in one of three scripting languages: JavaScript, C#, or Boo [23]. Unity simplifies programming by providing built-in visual effects such as particles, water, and post-processing. Artificial intelligence routines such as

Fig. 2.4 Sort 'Em Up, a serious game created by Squabble Studios to teach children about recycling [53]

path finding and steering behaviors are also provided. These routines allow amateur programmers to easily create believable non-player characters (NPCs) [58]. A single-player game can be converted into a multiplayer experience with minimal changes to the existing code. Unity also allows for easy integration with social media sites such as Facebook. Massively multiplayer games can be created by way of third-party solutions that must be purchased at additional cost through the Unity Asset Store [58]. Unity's asset store also contains many sound effects, images, 3D models, scripts, and complete games that can be used as a "starting point". However, the majority of the content found in the asset store is not free of charge. Community members are encouraged to upload assets to the store and set their own prices. Members receive 70 % of the sale price for each item sold through the store (Unity Technologies receives a 30 % commission on all sales).

Support is primarily provided by community members through online forums. A Premium Support package may be purchased granting the user direct access to the support team at Unity Technologies. Support packages cost between $500 USD and $1,000 USD a month, and there are limits to the number of times support can be requested and the number of users within a development team who may request support [58]. Unity does not charge on a per-title basis, and there are no annual maintenance fees. A free, Personal Edition of Unity is available that has a limited subset of available features. The Personal Edition of Unity can be used to create commercial projects royalty free provided that the licensee does not receive revenue or funding exceeding $100,000 USD in the previous year [58]. Projects (games) developed with the free version can be upgraded once a Unity Pro license has been purchased. A Unity Pro license costs $1,500 USD and permits publication of content created using Unity on desktop computers and the web (browser-based content). An additional license must be purchased at a cost of $1,500 USD for each mobile device (iOS, Android, BlackBerry) one wishes to publish content for [58]. Licenses may be provided at a reduced cost for developers of serious games or non-commercial projects. The licensing fee for serious game developers is negotiated on a case-by-case basis by the Unity sales team. Prices for console licenses are not posted and must be negotiated. Licensing the Unity source code is also possible.

2.2.2.2 Adobe Flash

Adobe Flash (formerly Macromedia Flash) began as a multimedia platform allowing 2D vector animations to be played in a web browser or as standalone applications. The free Flash player is included pre-installed on most computers and many mobile devices. It is estimated that the Flash player is currently installed on more than 1.3 billion computers and more than 500 million portable devices worldwide [2]. The Flash Builder framework combines a simple scripting language (Action Script) with a graphical editor used to position and animate objects. With Flash Builder, developers are able to easily release games on a variety of platforms including mobile devices. There are currently more than 50,000 mobile games and

applications developed using Flash [2]. Some of the more notable Flash games include Machinarium, a version of Angry Birds, and FarmVille. At its peak, FarmVille alone attracted more than 80 million active monthly players [9]. Flash now supports High Definition (HD) video playback, full screen viewing, and 3D GPU accelerated rendering [2].

Support is primarily provided by other community members through the online forums. Expert support packages are available for purchase from Adobe. There are also several books specific to Flash development currently available including "Flash Game Development by Example [19]. Flash builder is available for Windows and Mac OS. The content created with Flash Builder can be viewed on Windows, Mac OS, Linux, iOS, and Android. The Flash Builder software (standard edition) sells for $249 USD. Games created with Flash Builder can be sold without incurring additional license fees [2].

Plug-ins such as Flash are no longer needed to create rich interactive content online. The HyperText Markup Language (HTML) is the standard language used to create web pages. Until recently, the HTML standard did not support real-time interaction. Interactive elements and other forms of multimedia were added to web pages using web browser plug-ins such as Flash. The new HTML5 standard now includes video and audio playback, vector animation, and support for WebGL applications which allow for hardware-accelerated 3D graphics [56]. However, browser-based game development frameworks such as Flash may persist based on the strength of their development tools and a large user base.

2.2.2.3 Second Life

Second Life was launched in 2003 by Linden Labs, located in San Francisco, CA [33]. Second Life is not a game, and not a game engine. Instead, it is a massively multiplayer (supports a large number of players simultaneously) 3D virtual world; a social space where people can meet, chat, play, explore, and also construct their own virtual spaces within the world. Each Second Life account comes with an avatar that the player can customize to look any way they want. Players interact with each other and the world through their avatars. The content for this virtual world is created by the people who inhabit it. Massively Multiplayer Online games (MMOs) such as World of Warcraft (WOW) or EVE Online, provide a themed environment created by the game's developers. The virtual world within an MMO can be viewed as a stage where the game's narrative plays out. Second Life does not provide any theme or narrative. Instead, users are encouraged to create their own themed environments and items, including games [29]. The environments they create may be static or interactive. Users may use their creations to tell a story, or simply provide a space to inhabit and explore. Second Life has managed to harness the creativity of its user base in order to provide an ever expanding world that is as diverse as those who author it.

In Second Life, simple tools are provided for content creation. Original objects can be made by combining many simple shapes such as cubes, spheres, cones, etc.

[29]. Second Life also allows users to import models created with professional 3D modeling software such as Maya or 3D Studio Max. Objects in Second Life can be made interactive with the addition of scripts written in Linden Script (LSL), Second Life's proprietary scripting language. Scripted objects can move, react to being touched, and even respond to chat messages [21].

Linden Labs grants the users with intellectual property rights for the objects and environments they create, implying that users can buy and sell virtual items such as land, buildings, vehicles, fashion accessories, and even in-game pets [21]. Second Life employs its own currency called Linden dollars that can be bought or sold in exchange for real dollars. The rate of exchange fluctuates as it is based on what users are willing to pay. For example, between February 2008 and February 2011, the exchange rate ranged between 250 and 270 Linden dollars for one USD [32].

A standard membership is free of charge. A premium membership costs $9.95 USD per month. Premium members receive 1,200 Linden dollars (L$) a month [48], have the right to purchase land at auctions, and can have a second avatar on the same account [46]. A full region of land measures 65,536 m^2 and costs 295 L$ per month in addition to the purchase price. Land may also be purchased from Linden Labs or other users in smaller sizes. The price of virtual items in Second Life varies and is driven by the market [24].

It has been many years since Second Life was launched, and it currently maintains a large user base with new content added daily. As of June 2011, there were approximately 800,000 logins per month [32], and the average number of users online at any given time was estimated to be around 50,000 [24]. Many colleges and universities around the world have a presence in Second Life, often enabling students to explore a virtual version of the campus. Simulation Linked Object Oriented Dynamic Learning Environment (SLOODLE) is a free and open source project which integrates Second Life with Moodle, a learning management system. With SLOODLE, students can register for classes, attend lectures, and write quizzes within a virtual environment [51]. Real-world tourist attractions such as museums and historical sites are replicated in Second Life to educate and encourage tourism. Government agencies such as the Center for Disease Control (CDC) and the National Aeronautics and Space Administration (NASA) also have a presence in Second Life, with the goal of educating the public, and encouraging an interest in the sciences among students (see Fig. 2.5 for an example).

With Second Life, there are technical limitations due to the underlying architecture which at times could hinder its use for educational purposes. For example, the inclusion of user-created content forces the Second Life server to send every user's avatar and objects to every other user present in the same area. If too many users and unique objects are grouped together, it will lead to an increase in both computational requirements and network traffic (bandwidth) [60]. In addition to reducing immersion, lag caused by excessive network traffic reduces the accuracy of movement within the game and can lead to motion sickness [20]. Therefore, Second Life is not well suited to serious games that require accurate movement, pointing, or timing. Furthermore, the Second Life server is provided by and maintained exclusively by Linden Labs. Institutions may set up an "island"

Fig. 2.5 Large rockets on display at Spaceport Alpha in Second Life [52]

(a privately owned area of land within Second Life where the owner has control over the content and access), on the Linden server. In January 2007, Linden Labs released source code of the Second Life client software (Second Life Viewer). The Open Simulator community was able to create their own server software capable of connecting to Second Life clients [40]. Institutions can now host their own virtual worlds similar to those found on Second Life but with fewer restrictions. Second life restricts the number of objects and the size of scripts depending on the size of the land owned [30].

2.2.2.4 eAdventure

eAdventure was developed as a research project at Complutense University of Madrid (Spain), one of the oldest universities in the world [12]. eAdventure is the only framework in our top-ten survey results that was designed specifically for educational use. The framework has been designed to allow educators to author their own interactive content. eAdventure games can be integrated with learning management systems such as Moodle and Sakai [17].

Unlike most of the frameworks revealed by our literature survey, eAdventure is limited to building one specific type of game, 2D point-and-click adventures. Most step-by step procedures can be easily adapted to a point-and-click style interface.

By limiting development to one type of interaction, the framework is far less complicated than a general purpose game development tool. Games can be developed using the eAdventure language or created with a graphic-based editor which does not require a technical background or any programming skill [57]. eAdventure games can be deployed as a standalone application or as an applet playable in a web browser [17]. eAdventure is freely available and open source. You may use it, modify it, redistribute it, or integrate it into your project even if your project has commercial applications [17]. Documentation and tutorials are provided on the official site. Support is provided by the developers as well as community members by way of an online forum.

2.2.2.5 Unreal

The Unreal Engine was created by Epic Games, Inc., currently based in North Carolina, USA. Unreal Engine 1 was released in 1998 and since then, Unreal has been used to create more than 50 top-selling games, including Gears of War, Borderlands, Mass Effect, and BioShock [18]. In addition to being one of the most widely used commercial game engines, the Unreal engine has also been used to develop many well-known serious games, including the US Department of Defense's America's Army [59] (Fig. 2.6), and NASA's Moonbase Alpha [38]. Unreal Engine 4 features a 64-bit color HDR (High Dynamic Range) rendering pipeline with a wide range of post-processing effects, such as motion blur, depth of field, bloom, and screen space ambient occlusion. Unreal is well known for having some of the most realistic lighting effects, including real-time global illumination and surface interreflections [18].

Fig. 2.6 America's Army serious game [59]

Epic games now provides full access to the Unreal engine's C++ source code. However, most games are developed completely using UnrealScript, a proprietary object oriented scripting language that is similar in syntax to Java. Unreal also provides a visual scripting tool called Kismet that allows events and actions to be connected together by way of a "drag and drop" style interface. Kismet enables level designers to create simple behaviors and gameplay without writing any code. The Unreal Editor (UnrealEd) is a WYSIWYG editing tool. Designers can make changes to the environment and then preview the level directly within the editor. UnrealEd is more than just a level editor; it is a collection of visual tools for sculpting terrain, mesh painting, sound editing, amongst other operations [18]. One common complaint about the Unreal engine is that it is geared toward FPS style games. However, the Unreal engine can be used (and has been used) to develop many different types of games from virtually every genre even though game development with the Unreal engine becomes more difficult the farther you stray from the engine's FPS roots [61].

Support is provided by the large community of users consisting of both professional and amateur developers who provide help to fellow members in the community forums. Licensed developers receive a greater level of support provided by Epic [41]. The Unreal Development Kit (UDK) is free for non-commercial and educational use. For commercial projects, there is a 5 % royalty charged for revenue in excess of $3,000 USD per product per quarter [18].

2.2.2.6 Torque

The technology behind Torque Game Engine was originally developed by Dynamix for the Tribes series of games. Torque is currently maintained by GarageGames and has been re-engineered as a general purpose game-development framework. The Torque game engine was written in the C++ programming language and is now open source. The TorqueScript scripting language is similar syntactically to C++. Most of the game engine's functionality is available through TorqueScript, which makes it possible to build an entire game without the need to recompile the engine. The World Editor is a collection of graphical editors which allow artists to sculpt terrain, lay roads and rivers, place objects, edit materials, and create particle effects without writing any code [22].

Torque 3D supports Windows and all major web browsers. Unofficial support for other platforms may be provided by the community. GarageGames also offers a 2D version of the framework (Torque 2D) which is dedicated to 2D game development and supports most mobile devices. Both Torque 3D and Torque 2D are free under the MIT open source software license. Support is generally provided by the online community. Expert on-site support provided by GarageGames starts at $5,000 USD [22].

2.2.2.7 Ogre

According to our survey, Ogre was the 7th most utilized engine despite the fact that Ogre is a graphics engine only. In other words, if you want to add features such as audio, physics, and artificial intelligence to your game, you must either program such features yourself or integrate third party libraries into the project. Example programs combining Ogre with various physics and audio libraries are freely available thanks to the active community [39]. Ogre is freely available and open source under the MIT license. Supported platforms currently include Windows, Linux, Mac OSX, Android, iOS, and Windows Phone. Support is provided by the community via online forums [39].

2.2.2.8 Cocos

Cocos is a very popular freely available open source game development framework that supports development on all major mobile and desktop platforms. Cocos games can be deployed as a browser-based or stand-alone application [25]. Cocos includes visual editors for building the user interface, animating characters, and placing objects in the scene. There are several versions of Cocos being maintained by the community, some of which are focused on developing games for a specific platform. The most popular is Cocos2D-x which is written in C++ and supports the Javascript and Lua scripting languages. There are versions that support Python, C#, and Objective-C (iOS development) [10]. Cocos3D extends the 2D version by adding full support for 3D rendering based on OpenGL [11].

2.2.2.9 XNA

XNA Game Studio is game engine developed by Microsoft with the goal of making it easier to develop games for Windows, the Xbox, and Windows Phone [35]. It is free to develop XNA games for Windows, Xbox 360 game development requires a small yearly subscription fee. XNA was released in 2006 and quickly attracted a large community of independent game developers because of its simplicity and low cost [28]. Both 3D and 2D games can be created with XNA. XNA games written using the C# programming language. Although there are no official visual editing tools included with XNA, third party editors for UI design and level editing are available. Despite its popularity, Microsoft decided to retire XNA in 2014 [44]. The open source community has picked up where Microsoft left off with the creation of MonoGame. MonoGame is an open source implementation of XNA that has been extended to support additional platforms such as Linux, iOS, Android, and MacOS [37].

2.2.2.10 RPG Maker

RPG Maker was created by Enterbrain located in Tokyo Japan, and as the name suggest, this framework was designed specifically for the development of Role Playing Games (RPGs) only. More specifically, RPG Maker can only be used to create tile based games with separate combat screens viewed from the side (non-tiled). The developer is provided with limited control over the interface or game play, and must therefore focus on crafting a story told through NPC dialogs and quests instead. However, RPG Maker's singular focus makes it one of the simplest frameworks to use. No knowledge of programming is required. The visual editor assists the user in developing animated characters, and the WYSIWYG map editor allows trees, mountains and other objects to be added simply by selecting the object and clicking on one of the grid squares [45].

2.3 Discussion and Conclusions

Two literature surveys were conducted using Google Scholar, ACM Digital Library, and the IEEE Xplore Digital Library. The first survey (Survey 1) compared the number of papers mentioning the name of each engine in relation to search terms such as "serious game," "educational games," and "simulator." The frameworks and engines that scored well in this survey are those that are often discussed in academic publications. It could be argued that some of the frameworks listed here are not game engines or game development frameworks at all. Adobe Flash for example, is a framework for developing web-based applications in general, and is commonly used to make online advertisements. Second Life is a massively multiplayer virtual world, and it is not technically a game or a game engine. However, the content creation tools available to Second Life users, allows them to create their own games and interactive content. The serious games created with Second Life are essentially modifications ("mods") that exist as a location within the Second Life world. After Unity, Adobe Flash, Second Life, Unreal, and XNA Game Studio were referenced most often in academic publications. The second survey (Survey 2) attempted to discover which game engines and frameworks were used most often to develop serious games. A search was performed using the search term "serious game." Papers detailing the creation of a serious game were downloaded and read in order to determine which engine was used. Four of the top ten engines were full-featured game creation frameworks and six of the top ten were commercial products. However, three of these commercial game engines were free for non-commercial use. Out of the top ten most utilized frameworks, eAdventure was the only one designed specifically for educational purposes.

Table 2.1 summarizes some of the most popular game engines and frameworks (as determined by the survey results detailed in Sect. 2.1) along with common features that are important to developers of serious games.

Table 2.1 Popular engines/frameworks and their features (features that are available in the form of an add-ons created by a third party or with additional licensing fees are not included)

	Unity	Flash	Second life	e-Adventure	Unreal	Torque	Ogre	Cocos	XNA	RPG maker
Level editor	■	■	■	■	■	■		■		■
Scripting	■	■	■	■	■	■		■		■
C++					■	■	■	■		
Networking	■	■	■		■	■		■	■	
3D Graphics	■		■		■	■	■	■	■	
Shader effects	■				■	■	■	■	■	
Dynamic shadows	■				■	■	■		■	
Physics	■		■		■	■		■	■	
Artificial intelligence	■				■	■				
Free non-commercial	■		■	■	■	■	■	■	■	
Free for commercial				■			■	■	■	
Mobile devices	■	■			■		■	■	■	■
Web player	■	■		■				■	■	■

Companies such as Unity Technologies and Epic refer to their products as "game engines" even though they provide a complete set of visual editing tools. Adding to the confusion, frameworks such as Unreal include the visual editors with shipped games. This allows the modding community to modify and extend the game by adding new content. Free content created and shared by modders helps to extend the life of commercial games, which in turn leads to increased sales [27]. Many serious games are really game mods which can be distributed freely, although they do require the player to own a copy of the original game in order to play them. Graphics engines (APIs dedicated to rendering graphics only) are often referred to as game engines even though they lack much of the functionality required to develop a complete game. Part of the confusion stems from the fact that popular graphics engines are often combined with physics and audio engines by the user base, and distributed via personal webpages or as official tutorials. Academic publications to date rarely make the distinction between game development frameworks that supply a complete set of editing tools and game engines which supply a code base of core functionality.

The results presented here suggest that serious game developers are primarily using game engines and frameworks that were designed for the creation of leisure-based or entertainment-based games. Given the disparity in available resources to serious game developers when compared to commercial (entertainment) game companies, it is peculiar that they chose the same tools. This suggests that the currently available serious game engines may be lacking many of the features found in commercial engines. The results may also indicate that game engines designed specifically for serious games do not simplify the process beyond that of commercial entertainment focused engines and frameworks. Each game engine/framework has a number of features that may lend itself to a specific serious game development project.

It should be noted that although a significant amount of time and effort was placed to conduct the searches and review the hundreds of resulting papers, there are limitations to our results. More specifically, our results are specific to papers within three academic databases (Google Scholar, IEEE Xplore Digital Library, and ACM Digital Library), and therefore, game engines and frameworks that are not described in any research-based publications may have been missed altogether. The results of Surveys 1 and 2 are also subject to any bias Google Scholar, IEEE Xplore Digital Library, or the ACM Digital Library may have relating to the ordering of the search results. Furthermore, although access to the ACM and IEEE Digital Libraries was available, 46 % of the articles revealed in the Google Scholar search were not freely accessible and thus not considered.

Therefore, significant limitation was introduced by the fact that papers were selected only if they could be freely accessed. Despite these limitations, the Surveys 1 and 2 do help to remove any personal bias on the part of the authors in selecting the frameworks that warrant further investigation. Finally, although no claim is made that the conducted search is completely exhaustive of all serious games developed and all available development tools, a number of popular and commonly used game engines and frameworks, in addition to some of their features, have been outlined.

2.3.1 Content Experts and Educators as Game Developers

Most of the frameworks revealed by our surveys are complex game development tools requiring extensive training or programming knowledge before they can be utilized effectively. Content experts and educators often need to hire skilled developers to create games on their behalf. The eAdventure framework is unique in that it is targeted to educators who are not expected to have a game development background or any programming knowledge. Many educators are prolific in the amount of educational content they author for their students. This content takes the form of PowerPoint presentations, instructional documents, quizzes, videos, and websites. However, the educational content currently being created by educators is rarely interactive.

Educators could take on a greater role in the design and development of educational software and serious games if tools were made available to them that greatly simplified the game development process. The availability of such tools coupled with an educator's educational background (instructional design in particular), may lead to more effective serious games as well as the proliferation of interactive educational content. Although it is unrealistic to expect educators to be highly skilled in game development and programming, one can expect that some educators would be willing to take the time to learn some basic skills in order to provide a better learning environment for their students. If a game creation framework could be made "simple enough", a layperson could become competent enough to develop their own games, or at least modify existing games developed by others using the framework. eAdventure may be a step in the right direction, but

currently it is limited to the creation of one specific genre of game, and more specifically, 2D point-and-click adventures. Although 2D point-and-click adventures can be applied to a large number of learning applications, we believe that there is need for a simple general purpose framework for educators, or perhaps a series of frameworks each simplified for the development of one specific type of game, to ultimately provide educators (and serious games developers alike), greater freedom and flexibility.

Empowering content experts and educators to build their own interactive software and serious games may allow them to better meet the needs of their students. It could be argued that educators who do not have a background in game development may create games that do not offer an enjoyable experience for the players (students). However, it could also be argued that game developers who do not understand the educational content of the game or the needs of the students, may create games that do not meet the educational needs of the students.

To empower educators and provide them the opportunity to have a greater role within the serious game development process, we have assembled an interdisciplinary group of researchers, educators, computer scientists/engineers, and game developers, and have begun development of a novel game development framework that will greatly simplify the three primary areas of game development (art and animation, sound and music, and programming) [14]. The framework (known as the Serious Game Surgical Cognitive Education and Training Framework (SCETF)), allows for the creation of serious games by those who are not necessarily game developers or expert programmers (e.g., many educators). Although the framework may be of value to educators in general, we are currently targeting medical educators specifically. Serious gaming is growing in popularity within the medical education and training realm for a number of reasons including their ability to motivate and engage the trainees, and their cost effectiveness. However, medical procedures and practices are constantly changing in response to new technologies, new treatments, and a shifting patient demographic. By providing medical educators with the tools necessary to easily develop and/or easily modify their own interactive content (i.e., serious games), we are aiming for a proliferation of serious games for medical education and training with a current focus on cognitive-based surgical skills.

As part of the development of the SCETF, a needs analysis is being conducted to ensure that the design of the tool will meet the needs of the end user. This includes conducting a survey to gauge the technical background and computer literacy of clinical educators in order to gauge whether they possess the appropriate skillset and willingness to develop interactive software and games (given appropriate development tools). The results of this survey will then be used to guide the development process and ultimately allow us to strike a balance between ease of use and functionality, while providing the user with the necessary options and tools to ideally develop effective serious games. The survey itself has been evaluated by a panel of experts using the Delphi method to determine whether the survey questions are appropriate given the demographic and research goals. The Delphi method seeks to obtain a consensus on the opinions of experts, termed panel members,

through a series of structured questionnaires [16]. As part of the process, the responses from each round are fed back in summarised form to the participants who are then given an opportunity to respond again to the emerging data. The Delphi method is therefore an iterative multi-stage process designed to combine opinion into group consensus. Experts from a variety of fields; medicine, education, and computer science, have taken part in the panel.

Although participants for this survey are still being sought, preliminary results suggest that clinical educators generally do possess a high degree of computer literacy and are interested in developing their own interactive content. Participants have also indicated that the time they are able to devote to learning to use such tools is limited. The framework should therefore be made simple enough that an educator can learn to use it by following the provided examples in their spare time over a one to two week period.

In summary, serious games provide a viable education and training option, and are becoming more widespread in use for a variety of applications. However, in contrast to traditional entertainment game design and development, serious games and serious game designers/developers must strictly adhere to the corresponding content/knowledge base, while ensuring that their end product is not only fun and engaging, but is also an effective teaching tool. Despite the importance of proper serious game design, very few tools that are specific to serious game development and account for the unique requirements and complexities inherent in the serious game development process are currently available. In this paper we have provided insight into more popular tools currently used to develop serious games. We have also provided details regarding our current work that is seeking to bridge the game between what is currently available and what is actually needed by developers of serious games. Although plenty of work remains to be done, we are confident that these first steps will at the very least bring light to this important topic and ideally lead to greater work in the area of serious game framework and engine design.

Acknowledgements This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Social Sciences and Humanities Research Council of Canada (SSHRC), Interactive and Multi-Modal Experience Research Syndicate (IMMERSe) initiative.

References

1. Abt CC (1975) Serious games. Viking Compass, New York
2. Adobe Flash Product Information. http://www.adobe.com/ca/products/flash-builder-family.html. Accessed 27 Aug 2015
3. Arnab S (ed) (2012) Serious games for healthcare: applications and implications: applications and implications. IGI Global
4. Becker K, Parker J (2011) The guide to computer simulations and games. Wiley, Indianapolis
5. Bergeron B (2006) Developing serious games. Thomson Delmar Learning, Hingham, MA
6. Bishop L, Eberly D, Whitted T, Finch M, Shantz M (1998) Designing a PC game engine. Comput Graph Appl IEEE 18(1):46–53
7. Breuer J, Bente G (2010). Why so serious? On the relation of serious games and learning. Eludamos. J Comput Game Cult 4(1):7–24

8. Campbell J, Core M, Artstein R, Armstrong L, Hartholt A, Wilson C, Birch M (2011) Developing INOTS to support interpersonal skills practice. In: Aerospace conference, 2011 IEEE. IEEE, pp 1–14
9. Chiang O (2010) FarmVille players down 25 % since peak, now below 60 million. http://www.forbes.com/sites/oliverchiang/2010/10/15/farmville-players-down-25-since-peak-now-below-60-million/. Accessed 16 Nov 2015
10. Cocos2D game engine, Official website. http://cocos2d.org/. Accessed 13 Nov 2015
11. Cocos3D game engine, Official website. http://cocos3d.org/. Accessed 13 Nov 2015
12. Complutense University of Madrid, Home page. https://www.ucm.es/english. Accessed 23 Aug 2015
13. Corti K (2006) Game-based learning; a serious business application. PIXELearning, Coventry, UK
14. Cowan B, Kapralos B, Moussa F, Dubrowski A (2015) The serious gaming surgical cognitive education and training framework and SKY script scripting language. In: Proceedings of the 8th international conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp 308–310
15. Craighead J, Burke J, Murphy R (2007) Using the unity game engine to develop sarge: a case study. Computer 4552:366–372
16. Dalkey N, Helmer O (1963) An experimental application of the Delphi method to the use of experts. Manage Sci 9(3):458–467
17. eAdventure, Home page. http://e-adventure.e-ucm.es/. Accessed 22 Aug 2015
18. Epic games, unreal engine, Official website. http://www.unrealengine.com. Accessed 8 Nov 2015
19. Feronato E (2011) Flash game development by example: build 9 classic flash games and learn game development along the way. Packt Publishing Ltd
20. Fraser M, Glover T, Vaghi I, Benford S, Hindmarsh CGJ, Heath C (2000) Revealing the realities of collaborative virtual reality. In: Proceedings of the third international conference on collaborative virtual environments, New York, NY, USA, pp 29–37
21. Gans JS, Halaburda H (2013) Some economics of private digital currency. In: economics of digitization. University of Chicago Press
22. GarageGames, Official website for the Torque 2D and Torque 3D game engines. https://www.garagegames.com/. Accessed 8 Nov 2015
23. Goldstone W (2009) Unity game development essentials. Packt Publishing
24. Hill V, Meister M (2013) Virtual worlds and libraries Gridhopping to new worlds. Coll Res Libr News 74(1):43–47
25. Hussain F, Gurung A, Jones G (2014) Cocos2d-x game development essentials. Packt Publishing Ltd
26. Iuppa N, Borst T (2010) End-to-end game development: creating independent serious games and simulations from start to finish. Focal Press, Oxford, UK
27. Jeppesen LB (2004) Profiting from innovative user communities. In: Working paper, Department of Industrial Economics and Strategy, Copenhagen Business School
28. Keller B (2006) XNA studio: introduction to XNA. In: Game developer conference
29. Kemp J, Livingstone D (2006) Putting a second life "metaverse" skin on learning management systems. In: Proceedings of the second life education workshop at the second life community convention. The University Of Paisley, CA, San Francisco, pp 13–18
30. Konstantinidis A, Tsiatsos T, Demetriadis S, Pomportsis A (2010) Collaborative learning in opensim by utilizing sloodle. In: The sixth advanced international conference on telecommunications, 9–15 May, 2010, Barcelona, Spain, pp 91–94
31. Lewis M, Jacobson J (2002) Game engines. Commun ACM 45(1):27
32. Linden BK. Q1 2011 Linden dollar economy metrics up, Users and usage unchanged, June 5th 2011. Accessed 2013
33. Linden Labs, About Linden lab. http://www.lindenlab.com/about. Accessed 8 Nov 2015
34. Michael D, Chen S (2006) Serious games: games that educate, train and inform. Thomson Course Technology, Boston, MA

35. Miller T, Johnson D (2010) XNA game studio 4.0 programming: developing for windows phone 7 and xbox 360. Pearson Education
36. Mitchell A, Savill-Smith (2004) The use of computer and video games for learning: a review of the literature. London, UK. www.LSDA.org.uk
37. MonoGame, Official website. http://www.monogame.net/. Accessed 10 Nov 2015
38. NASA, Moonbase Alpha. http://www.nasa.gov/offices/education/programs/national/ltp/games/moonbasealpha/index.html. Accessed 1 Mar 2014
39. Ogre, Official website for Ogre, An open-source 3D graphics engine. http://www.ogre3d.org/. Accessed 12 Nov 2015
40. Open simulator, Information page. http://opensimulator.org/wiki/Main_Page. Accessed 14 Oct 2015
41. Petridis P, Dunwell I, Panzoli D, Arnab S, Protopsaltis A, Hendrix M, de Freitas S (2012) Game engines selection framework for high-fidelity serious applications. Int J Interact Worlds
42. Ratan R, Ritterfeld U (2009) Classifying serious games. In: Ritterfeld U, Cody M, Vorderer P (eds) Serious games: mechanisms and effects. Routledge, New York/London
43. Roman PA, Brown D (2008) Games–just how serious are they. In: The interservice/industry training, simulation & education conference (I/ITSEC), vol 2008, no 1
44. Rose M (2013) It's official: XNA is dead. http://www.gamasutra.com/view/news/185894/Its_official_XNA_is_dead.php. Accessed 25 Sep 2015
45. RPG Maker, Official website. http://www.rpgmakerweb.com/. Accessed 9 Nov 2015
46. Rymaszewski M (2007) Second life: the official guide. Wiley
47. Sawyer B (2009) Foreword: from virtual U to serious games to something bigger. In: Ritterfeld U, Cody MJ, Vorderer P (eds) Serious games. Mechanisms and effects, pp xi–xvi
48. Second Life, Official website. http://secondlife.com/. Accessed 20 Oct 2015
49. Sherrod A (2007) Ultimate 3D game engine design & architecture. Charles River Media
50. Shute V, Ventura M, Bauer M, Zapata-Rivera D (2009) Melding the power of serious games and embedded assessment to monitor and foster learning. In: Ritterfeld U, Cody MJ, Vorderer P (eds) Serious games. Mechanisms and effects, pp 295–321
51. SLOODLE (simulation linked object oriented dynamic learning environment). http://www.sloodle.org/. Accessed 22 Oct 2013
52. Space today online. http://www.spacetoday.org/Rockets/SecondLife/SL_Spaceport_Alpha.html. Accessed 23 Oct 2013
53. Squabble studios, Sort 'Em Up project page. http://www.squabblestudios.ca/sort.html. Accessed 10 Oct 2013
54. Squire K, Jenkins H (2003) Harnessing the power of games in education. Insight 3(1):5–33
55. Susi T, Johannesson M, Backlund P (2007) Serious games—an overview (Technical report no HS-IKI-TR-07-001). School of Humanities and Informatics, University of Skovde, Sweden
56. The World Wide Web Consortium (w3C) HTML standard. http://www.w3.org/standards/webdesign/htmlcss. Accessed 26 Aug 2015
57. Torrente J, Moreno-Ger P, Fernández-Manjón B, Sierra JL (2008) Instructor-oriented authoring tools for educational videogames. In: 8th international conference on advanced learning technologies (ICALT 2008), Santander, Spain, 2008, pp 516–518
58. Unity, Official website. http://unity3d.com/. Accessed 10 Oct 2015
59. US Army, America's Army serious game. http://www.americasarmy.com/. Accessed 9 Oct 2013
60. Warburton S (2009) second life in higher education: assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. Brit J Educ Technol 40(3):414–426
61. Zielke MA (2010) The game engine space for virtual cultural training. The University of Texas at Dallas Arts and Technology