Chapter 5

Game Engine Solutions

Anis Zarrad

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71429

Abstract

The rapid development of hardware and system platforms provides a favorable foundation for game development. A game engine overview is introduced first. Then, key features and available solutions of game engines are discussed. Typical products of game engines are shown and evaluated. Finally, we summarize our findings.

Keywords: game engine, game builder, 3D game, 2D game, engine architecture

1. Introduction

Game engines are a new way to develop high-quality games easily and rapidly without needing intensive programming skills and computational resources. Today, there is growing interest in game engines due to the rapid development of hardware and system platforms.

3D game engines help game companies reduce their cost, time, and manpower, since game developers can use the available functionalities of the engine. However, with over 100 engines available in the market for commercial and educational purposes, with a wide range of diverse features, each with its own performance levels, license types, and cost structures, selecting an appropriate game engine for a specific purpose becomes a challenging problem.

Game engine systems hit the headlines only a few years ago. Before the appearance of game engine technologies [1, 2], existing systems were often developed as virtual augmented reality systems to handle specific tasks such as NPSNET [3], DIVE [4], and SPLINE [5]. Thus, any modification required a hard change in the programming environment and architecture. As game engine technology matures and becomes more flexible, implementing a 3D environment will become easier. Despite these improvements, programming skills remain a concern, and usually sabotage developers who want to create complex environments.

Game developers usually have advanced programming skills and often block the system from user manipulation or misuse, which further complicates the task. Most game engines are limited to specific tasks and their features are typically coupled with specific game characteristics. Thus, developing extensions or modifications that force a game engine to adopt a new class of applications is almost impossible. Recently, many techniques and programming approaches such as virtual reality modeling language (VRML) [6], OpenGL [7], DirectX [6], X3D [6], and MPEG-4 [8] have been developed to build game applications. However, many of these new methods cannot provide native support to extend existing games. Systems like Bamboo [9] and JADE [10] have been proposed to overcome the limitations, brought about by the interconnected gaming engine and development platform, to offer a better solution.

In this chapter, we evaluate the latest released game engine from a variety of aspects like modularity, performance, usability, library, speed, and the realism. The readers will obtain an overview of game engine research, while becoming familiar with the recent developments and technologies in this area.

2. Existing game engine solutions

As technology moves forward, the need for multiuser game applications is getting more attractive. Before game engines, games were typically written as singular entities. Any modification to a game system required a collaborative effort from graphic designers, game designer, and 3D programmers to develop the required new scenario. Then, a player would need to stop and restart the game to reflect the modifications. Today, most researchers seek to build a gaming platform that facilitates the development and modification of such applications easily and rapidly without having intensive programming skills. For this reason, a game engine concept, which separates the engine and content development, arose in the mid-1990s. By the end of 1998, many games like ID Software's Quake III [11] Arena and Epic Games' 1998 Unreal [12] were designed with a game engine approach in mind.

Currently, there are many modern 3D engine games such as Unity engine [13], Unreal engine [12], Gamebryo engine [14], CryEngine [15], and Software's Source engine [16]. Choosing an adequate game engine depends on the type of game you want to create and the platform you want to use.

Unity engine [13] is a well-known game engine developed by Unity Technologies in Denmark; it is used to create interactive 3D content using JavaScript. Unity does not offer as many features as other tools like Unreal and CryEngine. Therefore, making excellent games without having a full license, which can cost nearly $2000, is very difficult. Unity integrates a custom rendering engine with an NVIDIA PhysX physics engine [17].

The Unreal engine [12] offers a completely royalty-free version with a powerful rendering engine and environment editor. It contains a variety of application programming interface (APIs) and tools to let you create a 3D virtual environment that closely resembles the real world. Unreal comes with partial documentation, while Unity and Source engines provide a complete documentation with detailed examples. Unreal Engine uses C++ while Unity uses C# or JavaScript.

http://dx.doi.org/10.5772/intechopen.71429

Software's Source engine [16], developed by Valve Corporation, uses Half-Life 2 [18] and Portal. The overall system architecture is based on the popular modular architecture to provide independent updates. It uses an SDK source and a few GUI-based programs to handle complex functions.

Compared to the Unreal Engine [19] and CryENGINE [15], Emergent Game Technologies' Gamebryo engine [14] is used to develop a variety of games, ranging from action games to strategy games like Fallout 3, Empire Earth, and Civilization IV games. Gamebryo also supports a variety of platforms such as Windows, Wii, Xbox 360, and PlayStation.

CryEngine [15] is a game-development tool developed by Crytek Company. It facilitates event scripting, animation, and 3D object creation in the free CryEngine SDK. CryEngine is designed to support PC platforms and consoles, including Xbox 360 and PlayStation. Panda 3D [20] is a complete game and simulation engine with a very rich feature set. It uses advanced shaders and rendering effects, with different techniques. Torque 3D [21] is designed from the ground up to allow developers to make their game abstracted completely away from the underlying hardware. This represents a vast benefit for game developers, allowing them to work on game project without focusing on platform-specific requirements or restrictions.

Before the Game Engine, virtual environments were developed using dedicated systems to implement a specific scenario model. Some of the most well-known systems of this kind include DIVE [4], MASSIVE [22], NPSNET [3], SPLINE [5], and VLNET [23]. These systems focus on specific applications to reduce the overall implementation complexity. The major problems stem from the fact that systems are tightly coupled in terms of implementation. Consequently, any modifications in the application require modifications in the supporting architecture, which occurs because the complexity property is gained through the combination of the internal architecture with the specific application functionalities.

Bamboo [9] uses a microkernel-based architecture to separate the system elements from the kernel in such a way that the add, remove, and modify functions can be performed at runtime. Unfortunately, this highly accredited approach incurs a great deal of complexity, particularly in terms of facilitating communication between components written in different languages. Oliviera et al. [10] developed a Java Adaptive Dynamic Environment (JADE) based on Java architecture. It consists of a lightweight cross-platform kernel that permits system evolution at runtime. The adoption of JADE does not provide an efficient solution to problems that arise from extending virtual reality applications.

In [24], the authors developed the Virtual Environment Mark-up Language (VEML) based on the nonlinear story [24] concept defined by Szilas [25] to build virtual reality applications. This model allows the progress of the story, during the simulation, to be specified independently from the implementation of the 3D object geometry and from the 3D-environment programming.

3. Game engine architecture and principles

An engine is an essential part of a game; it influences the structure and the organization of game graphics, configuration files, and all other inputs such as user inputs, maps, and sounds. **Figure 1** shows a game engine component diagram in a game development context.

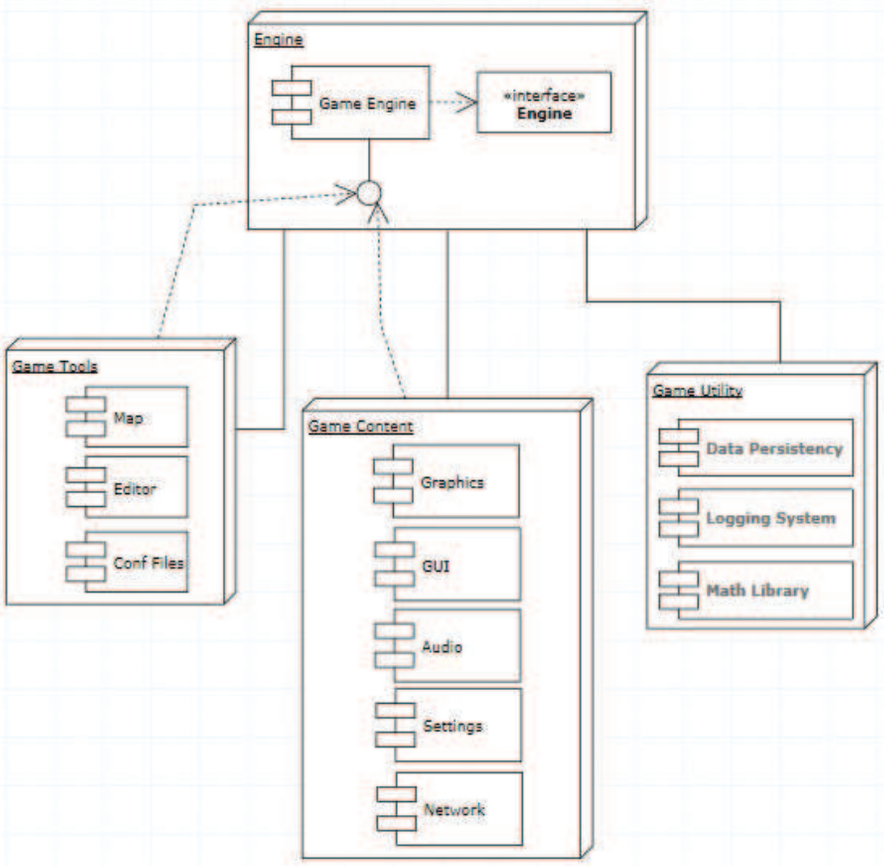

Figure 1. Game engine component diagram.

Game tools, Game Utility, and Game Content are also vital components of a game. Game tools refer to all the tools used to develop a game, such as the character editor, ability editor, configuration files, and game map. Game content refers to all related features in any game. Graphics and audio files are the most important type of data. Network components allow multiple users to connect to each other and access game data. Utility components define all needed data types, message formats, and required files, with respect to the game characteristics. The game tool component plays a bridge role between Game content and the Game engine components.

Currently, no specific standard architecture has been developed in the literature. **Figure 2** shows a detailed game engine architecture. We consider only some features that are most likely important.

Manager components are essential to all games. A physical manger is important for games dependent on physical reactions. For example, car racing needs physical laws, but this is not so important for a card matching game. However, the input manager and the GUI system are important, even for a simple game, to process the user input. The scene manager is required to organize and control the scene content and monitor the game's logic. The game loop is used to control an infinite loop that keeps the game running over-and-over again. The network component is responsible for managing user connections and game data access. The AI System is a special module designed and written by software engineers with specialized knowledge.

http://dx.doi.org/10.5772/intechopen.71429

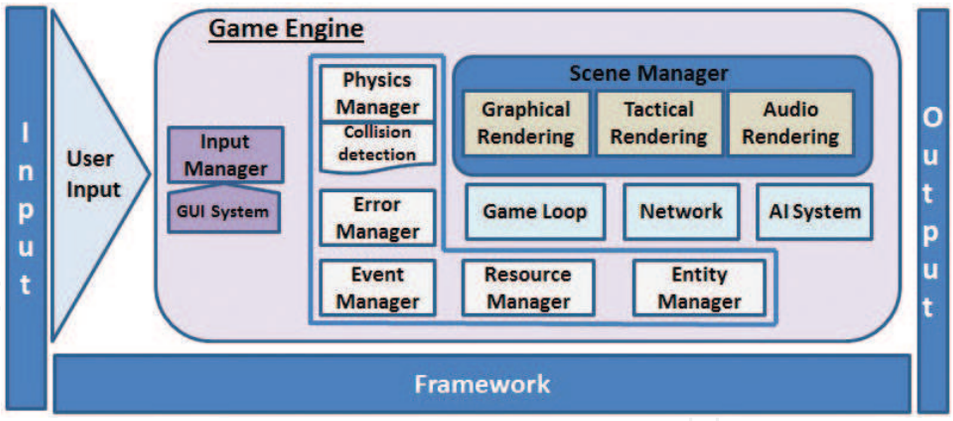

Figure 2. Detailed game engine architecture.

Additional manager components, which depend on programmer and coding approaches like game states, Rule engine, and Speech recognizer, may be needed at specific times.

Designing a game engine solution to fit all game purposes is a challenging problem for an engine developer. There is a need to define key design principles and provide advice to create successful engines. The five principles listed in this section and referred with the acronym MULER are modularity, usability, library resources, efficiency, rendering effects, and picture quality. Each of these features is detailed below and will be used to evaluate available existing popular game engines in Section 4.

3.1. Modularity

Game engine should be implemented via several unique modules. Each manager component is independent much like a single functional unit. Engine developers should invest time in implementing a modular game architecture to reduce the system complexity and offer robustness. Testing process and maintenance becomes easier for the programmer when needed. As shown in the proposed architecture in **Figure 2**, the principle of modularity is satisfied and each component has a unique function.

3.2. Usability

Engine developers are the main actors; therefore, usability of a game engine is an important criterion to evaluate. As defined by [26], usability includes ease of learning, efficiency of use, memorability, error frequency and severity, and subjective satisfaction. Ease of learning and efficiency of use are the main aspects of usability. We evaluated available tutorials, project examples, forum communities, programming languages, and technical support. Information is retrieved from the engine's website. For efficiency of use, we investigated game engine application editors including map editor, GUI editor, animation editor, programming editor, and scene editor. We considered the Debugger as part of the programming editor. This option helps a developer to stop applications at specific lines of the source code and check values. This is helpful for developers because they can easily find the error and fix it.

3.3. Library resources

3D game engine systems are implemented by assembling many resource libraries together. The engine developer selects a game engine based on its available programming resource libraries. A common library for a simple 3D game engine consists of 3D graphics, physics, collision detection, input/output, audio, AI, 3D graphics, and a network library. Modern game engines can include more powerful libraries including physics laws, collision detection, and special effects. In addition, resources in the game engine contain project examples, user guides, and tutorials; these available resources are extremely useful for a game developer. We believe this is the most useful criterion for game engine comparison

3.4. Efficiency

Game engine efficiency refers to the successful use of all inputs and available resources to produce a given game output. It includes memory allocation, CPU usage, rendering process, and other features. We may produce an efficient engine but not an efficient game. This is because the game workflow is designed by the game creator, which has an impact on the game efficiency. Resource managers play an important role for this criterion. It loads the game data and files from disk into the memory to be used for rendering and game scenario creation.

Game designers should consider the reusability and modularity concepts to improve engine efficiency. Single resources can be used to create many instances.

3.5. Rendering effects and picture quality

The rendering process produces 3D animated graphics using specific techniques like rasterization [27], image-based rendering (IBR) [28], ray tracing [27], or any different technique. These techniques are valuable for 3D game engines.

Unlike 2D graphics, the implementation of a 3D graphics rendered requires advanced programming and 3D modeling skills. 3D graphics have countless rendering algorithms from extremely fast to extremely slow. Hardware and software rendering have an impact on the acceleration of the 3D graphics. Most available engines render a scene using the default settings. These features may be good for a quick preview, but not for production work. With modern engines, you can get a very sharp, clear, and high-quality output; this comes at the price of increased rendering time even though the overall final output will be much better.

4. Game engine evaluation

In this section, we evaluate the selected game engines based on the most important principles MULER: modularity, usability, library resources, efficiency, rendering effects, and picture quality. Currently, there are 376 game engines listed in the DevMaster.net database [29]. We set a game engine selection filter based on the following criteria:

http://dx.doi.org/10.5772/intechopen.71429

- Game engines that lack important features are excluded.
- No high-quality games are implemented.
- No recent update or released versions.

We studied all game engines listed in DevMaster. Only 20 game engines are selected for evaluation. **Table 1** shows the details of the selected engines.

Our goal is to provide a complete evaluation of game engines. Developers want to know which game engine to invest on. Selected engines will have an impact on the game output. **Table 2** will help users to pick the optimal 3D game engine based on their purpose and needs.

The evaluation criteria for each principle are defined as follows:

- Modularity: from "0" to "3" to indicate the modularity level. "0" means modularity is not considered in the game engine architecture.
- Usability: from "1" to "5" to indicate the usability level. "5" represents complete user satisfaction. Score assignment is based on user reviews and forum community responses.
- Library resources: limited resources, acceptable resources, very rich resources.
- Efficiency: poor, good, excellent.
- Rendering effects and picture quality: poor, good, excellent.

4.1. Discussion and general findings

Unity 3D has a poor usability because GUI and AI editor are not implemented. There is also no real demo or technical support provided in the last release. Unity 3D implements a very basic PhysX module, which affects the performance and rendering aspects of the system. Similarly, Jmonkey [30] has a poor performance for heavy games because NiftyGUI library used a JavaBuilder pattern for creating controls on the screen, this way preventing for subclassing elements, which renders the code unreadable. Many users complain about Jmonkey's "very slow engine." Nevertheless, the system-provided documentation covers almost every topic.

Shiva3D's engine can be deployed in more than 10 platforms. It is a quite powerful engine with many features. The dynamic and static shadow management on a custom textured model needs more enhancement, especially for real-time games. Marmalade engine has a poor rendering process since the engine uses limited animation and a mesh library.

One of the best engines available in the community is Unigine. It has a good performance for large-scale games. It also has a full-fledge rendering engine with the latest implemented features (full-dynamic lighting, DX11 tessellation). Recently, a 3D planet system with geographic coordinates is integrated into the engine. Rendering and physics features are comparable to engines like CryEngine and Unreal. However, Unigine's lack of tutorial and example availability is a negative. We cannot decide on the modularity score due to missing information.

Game engine	Language	Supported platforms	2D	3D	License/price
Unity 3D	C# and JavaScript	**Desktop**: Windows, Mac OS, Linux **Mobile**: Android, IOS,	No	Yes	Free for Indie version
Jmonkey Engine	Java, Python, and JavaScript	**Desktop**: Windows, Mac OS, Linux **Mobile**: Android	No	Yes	Free
Marmalade	C/C++	**Desktop**: Windows, Linux, **Mobile:** Android, and IOS	—	Yes	Commercial
LibGDX	Java	**Desktop:** Windows, Mac OS, Linux **Mobile:** Android, IOS, Blackberry, HTML5	Yes	Yes	Free
Panda3D	Python and C++	**Desktop**: Windows and Mac OS	No	Yes	Free/open source
Blender	C/C++, Python	**Desktop**: Windows, Mac OS, Linux	No	Yes	Free
Unreal engine	C++, VisualScripting	**Desktop**: Windows, Mac OS, Linux, HTML5 **Mobile:** NA			Free/open source
CryEngine	C/ C++	**Desktop**: Windows, and Mac OS **Mobile:** Android, and IOS,	No	Yes	Unspecified
Crystal Space	C/C++, Python	**Desktop**: Windows, Mac OS, Linux **Mobile:** NA	—	Yes	Free
CopperCube	JavaScript, Flash, WebGL	**Desktop**: Windows, Mac OS **Mobile:** Android	Yes	Yes	Free
Leadwerks	C++, C#, VB.Net, Python	**Desktop**: Windows **Mobile:** NA			$99.95
Raydium	C/C++	**Desktop:** Windows, MacOS, Linux **Mobile:** IOS	No	Yes	Free
SunBurn	C#	**Desktop:** Windows **Mobile:** Windows Phone	No	Yes	$ 150
UDK	C/C++	**Desktop**: Windows **Mobile:** iOS	No	Yes	$99.00
Corona SDK 61	Lua	**Desktop**: Windows, Mac OS **Mobile:** iOS; Android; Windows Phone	Yes	No	Free
3D Game Studio	C++, C#	**Desktop:** Windows **Mobile:** NA	Yes	Yes	Free for the basic version
Unigine	C/C++	**Desktop:** Windows, and Linux **Mobile:** iOS; Android	No	Yes	$ 25,000
Torque3D	C++, TorqueScript	**Desktop:** Windows, Mac OS, and Linux **Mobile:** NA	—	Yes	Free/open source
ShiVA	C/C++, Lua	**Desktop:** Windows, Mac OS, and Linux **Mobile:** iOS; Android; Windows Phone, and Blackberry	—	Yes	Free for personal use
Irrlicht	C++, C#, VB.Net	**Desktop:** Windows, Mac OS, and Linux **Mobile:** iOS; Android;	Yes	Yes	Free

Table 1. Details of the selected game engines.

http://dx.doi.org/10.5772/intechopen.71429

Game engine	Modularity	Usability	Efficiency	Library resources	Rendering effects and picture quality
Unity 3D	1	4	Acceptable	Acceptable resources	Acceptable
Jmonkey Engine	3	2	Poor	Rich resources	Acceptable
Marmalade	1	1	Poor	Limited resources	Poor
LibGDX	2	3	Acceptable	Rich resources	Acceptable
Panda3D	3	4	Acceptable	Acceptable resources	Excellent
Blender	3	5	Acceptable	Rich resources	Acceptable
Unreal engine	4	4	Excellent	Rich resources	Excellent
CryEngine	3	5	Excellent	Rich resources	Excellent
Crystal Space	5	1	Poor	Acceptable resources	Acceptable
CopperCube	2	3	Poor	Limited resources	Poor
Leadwerks	4	1	Excellent	Acceptable resources	Excellent
Raydium	1	1	Poor	Acceptable resources	Acceptable
SunBurn	1	2	Good	limited resources	Good
UDK	2	5	Excellent	Very rich resources	Excellent
Corona SDK 61	1	4	Good	Acceptable resources	Acceptable
3D Game Studio	1	4	Good	Acceptable resources	Poor
Unigine	—	4	Excellent	Very rich resources	Excellent
Torque3D	2	3	Good	Acceptable resources	Acceptable
ShiVA	3	3	Poor	Acceptable resources	Good
Irrlicht	—	2	Good	Acceptable resources	Good

Table 2. Game engine evaluation based on MULER.

Unreal engines handle shading in an efficient way by using a DirectX 11 pipeline that includes deferred shading, global illumination, lit translucency, and postprocessing. It also integrates NVIDIA technologies. The physics management component in the CryEngine is excellent. The game engine has built-in rendering paths for OpenGL and DirectX 8/9,

allowing to support Linux, Apple, and Microsoft operating systems. The various rendering paths also allow more hardware to be supported by the engine.

SunBurn has a basic physics module and only support windows and windows phone 7. It uses flexible rendering and provides both advanced deferred rendering and traditional forward rendering, allowing you to choose the ideal technique for your project. It does not have a network or AI library. Raydium does not offer a GUI editor, scene editor, and animation editor. It also does not have mapping or AI modules implemented.

UDK [31] is a complete professional development framework. It includes a set of features packed with power and ease of use. UDK has a very easy interface while offering power and flexibility. The rendering process is exceptional due to the use of a multithreaded rendering system. UDK makes the most beautiful graphic games in mobile devices because UDK can transform global illumination to texture.

In Panda 3D [20], the C++ documentation is lacking compared to the Python documentation. It uses C++ to solve the performance issue. Projects developed with Torque can be easily ported to other operating systems and platforms as explained by Nilson and Söderberg [32]. Blender [30] engine is built into an exceptional 3D modeler, which means that "importing" models into your game is as simple as playing it. The Python scripts and game logic are very powerful and easy to use. Blender has a layered architecture including utilities, kernel, computer, render, UI, and applications. LibGDX [33] is mainly developed for 2D mainly with cross-platform tool so you can make games for Windows, Linux, OS X, HTML, Android, and iOS. 3D game Studio engine offers allow developers to build a complete game without knowing C++. In addition, the engine has an integrated 2D engine. However, the engine cannot be used to developed large-scale game with excellent graphics quality.

Irrlicht [34] uses a low-quality rendering model. It supports a wide range of 3D/textures formats. Irrlicht has a very large and friendly community. CrystalSpace [35] requires Cygwin when compiling on windows. Thus, it is too hard for developers to set up modeler and management components. Compared to other engines, CrystalSpace and Jmonkey [34] seem to have the lowermost usability level.

Leadwerks has a very limited support that affects its usability principle. Performance is enhanced with the last release, Leadwerks Game Engine 5. It now decouples the game loop from the renderer. Similarly, usability in CopperCube is not satisfied, the option to drag and move objects into the space is not implemented, and JavaScript API should be extended to include more advanced features.

5. Conclusion

A comparison table shows that there is no best game engine for all games. Each game engine has its own advantages and disadvantages. Cleary Shiva3D and LibGDX are the best in terms of platform deployment criterion. It can be deployed in more than eight platforms. However, when we consider performance and rendering efficiency criteria, UDK, CryEngine, and Unigin are the clear winners.

If you are a developer with mixed skills, then try out UDK and Unreal Engine 4. Making a good game is not risked by choosing a wrong engine. They are just a tool. In my opinion, the best way to find your suitable tool is to play around with the engine and then decide.

Author details

Anis Zarrad

Address all correspondence to: anis.zarrad@gmail.com

Prince Sultan University, Riyadh, Saudi Arabia

References

[1] Cowan B, Kapralos B. A survey of frameworks and game engines for serious game development. In: Advanced Learning Technologies (ICALT): IEEE 14th International Conference on Greece; 2014

[2] Jacobson J, Lewis M. Game engine virtual reality with caveut. IEEE Computer. 2005; **38**(4):79-82

[3] Michael RR, Macedonia M, Zyda J, David R, Pratt R. NPSNET: A network software architecture for large scale virtual environments. Presence: Teleoperators and Virtual Environments (USA). 1994;**3**:256-287

[4] Hagsand O. Interactive MUVEs in the DIVE system. IEEE Computer. 1996;**3**(1):30-39

[5] Waters R, Anderson D, Barrus J, Brogan D, Casey M, McKeown S, Nitta T, Sterns I, Yerazunis W. Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability TR-96-02a November; 1996

[6] Xie W, Yanrong L. The virtual furniture store constrution based on VRML/X3D. In: 2012 International Conference on Computer Science and Information Processing (CSIP), China. 2012

[7] Lee H, Baek N. Implementing OpenGL ES on OpenGL. In: ISCE '09: IEEE 13th International Symposium on Consumer Electronics, Japan; 2009

[8] Battista S, Casalino F, Lande C. MPEG-4: A multimedia standard for the third millennium 1. IEEE MultiMedia. 1999;**6**(4):74-83

[9] Magerko B, Laird JE. Building an interactive drama architecture. In: 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Germany. 2003. pp. 24-26

[10] Oliveira M, Crowcroft J, Slater M. Component framework infrastructure for virtual environments. In: Proceedings of the third international conference on Collaborative virtual environments. USA. 2000. pp. 139-146

[11] Mamei M, Zambonelli F. Motion coordination in the Quake 3 arena environment: A field-based approach. In: International Workshop on Environments for Multi-Agent Systems, 2004. pp. 264-278

[12] Lewis J, Brown D, Cranton W, Mason R. Simulating visual impairments using the unreal engine 3 game engine, In: IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH), Portugal. 2011. pp. 45-53

[13] Polančeć D, Mekterović I. Developing MOBA games using the unity game engine. In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Croatia. 2017

[14] Emergent: Gamebryo Game Engine, Available from: http://www.emergent.net/

[15] Juarez A, Schonenberg W, Bartneck C. Implementing a low-cost CAVE system using the CryEngine2. Entertainment Computing. 2010;**1**:157-164

[16] Kovacs D, Mitchell J, Drone S, Zorin D. Real-time creased approximate subdivision surfaces with displacements. IEEE Transactions on Visualization and Computer Graphics. 2010;**16**(5):742-751

[17] Zhonghua L, Sankaranarayanan G, Deo D, Chen D, Suvranu D. Towards physics-based interactive simulation of electrocautery procedures using PhysX. In: IEEE Haptics Symposium, USA. 2010. pp. 125-137

[18] Cricenti L, Branch A. A generalised prediction model of first person shooter game traffic. In: IEEE Local Computer Networks, LCN 34th Conference, Switzerland. 2009

[19] Ferreyros C, Wendorf M, Juárez P. Developing a Videogame Using Unreal Engine Based on a Four Stages Methodology. Peru: IEEE ANDESCON; 2016

[20] Goslin M, Mine MR. The Panda3D graphics engine. Computer Journal. 2004;**37**(10): 112-114

[21] Shiratuddin F, Fletcher D. Development of southern Miss's innovation and commercialization park virtual reality environment. Proceeding Sixth Conference on Construction Applications of Virtual Reality, USA; 2006. pp. 278-285

[22] Benford S, FahlŽn L. A spatial model of interaction in large virtual environments. In: Proceedings of the 3rd conference on European Conference on Computer-Supported Cooperative Work, Italy. 1993. pp. 109-124

[23] Pandžić S, Tolga K, Elwin L, Thalmann N, Thalmann D. A flexible architecture for virtual humans in networked collaborative virtual environments. Proceedings Eurographics, Hungary; 1997. pp. 177-188

[24] Boukerche A, Duarte D, Araujo R, Andrade L, Zarrad A. A novel solution for the development of collaborative virtual environment simulations in large scale. Ninth IEEE International Symposium on Distributed Simulation and Real Times DS-RT Proceedings, Canada; 2005. pp. 86-97

[25] Szilas N. IDtension: A narrative engine for interactive drama. In: Proceedings of TIDSE'03. Frauenhofer: IRB Verlag; 2003

[26] Williges T, Hartson R. The effectiveness of usability evaluation methods: Determining the appropriate criteria. In: Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting, USA. 1999. pp. 1090-1094

[27] Davidovič T, Engelhardt T, Georgiev I, Slusallek P, Dachsbacher C. 3D rasterization: A bridge between rasterization and ray casting. Proceedings of Graphics Interface, Canada; 2012. pp. 201-208

[28] Lai J, Chen C, Chien Y. Architecture design and analysis of image-based rendering engine. In: 2011 IEEE International Conference on Multimedia and Expo (ICME), Spain

[29] DevMaster.net. Game Development. 2017. Available from: http://devmaster.net

[30] Navarro A, Vicente Pradilla J, Rios O. Open source 3D game engines for serious games modeling. Book Chapter ISBN 978-953-51-0012-6, March, 2012

[31] Shiratuddin M, Thabet W. Virtual office walkthrough using a 3D Game Engine. International Journal of Design Computing. 2002;**4**(10):24-28

[32] Nilson B, Söderberg M. Game engine architecture. Mälardalen University 2007. Available from: http://www.idt.mdh.se/kurser/cd5130/jgms/2007lp4/report9.pdf

[33] Scacchi W. Practices and technologies in computer game software engineering. IEEE Software. 2017;**35**(1):110-116

[34] Rocha RV, Rocha RV, Araújo RB. Selecting the best open source 3D games engines. In: Proceedings of SBGames. 2010

[35] DeChiara R, Erra U, Andreoli R, Scarano V. Interactive 3D environments by using videogame engines. In: 17th International Conference on Information Visualisation. Vol. **1**(2). 2013. pp. 515-520

IntechOpen