# Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices

Eleftheria Christopoulou, Stelios Xinogalos
*Department of Applied Informatics, University of Macedonia, Greece*
*{mai1519, stelios}@uom.edu.gr*

## Abstract

*Game engines are tools that expedite the highly demanding process of developing games. Nowadays, the great interest of people from various fields on serious games has made even more demanding the usage of game engines, since people with limited coding skills are also involved in developing serious games. Literature in the field has studied game engines focusing on specific needs, such as 3D mobile game engines or open source 3D game engines. The motivation of this article and at the same time the advancement brought by it in the field, lies in the extension of an existing framework for the comparative analysis of several game engines that export games at least on Android and iOS mobile devices and cover a wide range of different user profiles and needs. In order to validate the results of this comparative analysis a shooter game was developed for Android devices based on official tutorials of the two game engines that came out to be more powerful, namely Unity and Unreal Engine 4. In conclusion, there is not a single game engine that is better for every purpose and the extensive overview provided can help users choose the most suitable game engine for their needs.*

**Keywords:** *Game Engines, Comparison, Mobile devices, Android, Unity, Unreal Engine 4*

## 1. Introduction

Nowadays, serious games as well as mobile games development are on the rise. Day by day, desktop and console games are replaced by games which run on mobile phones and tablets. Moreover, people involved in the design and development of serious games come from various fields. For example, pedagogists and domain experts with limited, if any, coding skills are involved in designing and developing serious games. So, the choice to develop a game by using solely a programming language, such as C++, C# or Java, is not really feasible for everyone. This has made the usage of game engines even more important. Game engines expedite the process of developing a game through existing templates and assets that can be reused, minimizing or completely extinguishing the need to have a deep knowledge of programming. Moreover, game engines give the chance to develop a game once and export it to various platforms, including mobile devices, by making just a few changes to the original version.

However, there are many game engines that share some common features but also have a lot of differences. Actually, the various game engines have a different philosophy on game development and aim at a wide range of different needs: engines that do not require knowledge of programming; engines that are based on popular web technologies; engines that are open source and can be customized/extended by experienced users; and professional game engines. Making an informed choice of a game engine is a multi criteria decision and is not easy. Although there are several review articles on game engines, in most cases these articles focus on a specific type of game engines (e.g. 3D mobile game engines) or a specific domain (e.g. simulated surgical training). Providing an overview of game engines that cover a wide range of different user profiles and needs based on an extended list of features is considered important. To the best of our knowledge there is no such overview and this provided us the motivation for this article. Consequently, the aim of this article is to inform anyone interested about: all the features that might be present in a game engine organized in higher level categories; the main types of game engines; the capabilities offered by representative examples of game engines falling into the

aforementioned categories. The ultimate goal is to provide support in selecting the most suitable game engine based on the needs of a specific game project and the profile of the person(s) that are going to implement it.

The rest of the article is organized as follows. In section 2 related work on evaluation and/or survey and comparative analysis of game engines is summarized. In section 3 the game engines selected for the comparative evaluation are briefly described. In section 4 the framework used for our comparative analysis of game engines is briefly described along with the amendments that were made and the methodology used for preparing the necessary information and presenting the results. The results of the comparative analysis are presented in section 5. In section 6 we briefly present the results of an empirical study that refers to utilizing the two most powerful game engines –based on the results of the comparative analysis- for implementing a simple Android shooter game. Finally, in section 7 we present the conclusions of the study presented in this article and recommendations for future research.

## 2. Related work

Comparative analysis and evaluation of game engines is a contemporary open issue in game development. The constantly increasing interest of a wide range of stakeholders for serious games has made this comparative analysis and evaluation of game engines even more important. People with different or no background at all on game technology are nowadays interested in developing serious games. Being aware of the various types of game engines and their features can help potential game makers to make informed decisions on selecting the appropriate game engine based on their goal and personal skills. In this section we review related work and conclude with the main goal of our study.

Andrade [1] conducted a survey of various games engines, providing a brief presentation of their features and a comparative analysis in tabular form.

Pattrasitidecha [2] in his master thesis compared and evaluated various 3D mobile game engines. The result of this work was a comparison of the aforementioned type of game engines in a tabular form aiming at supporting readers in choosing a game engine based on their needs. A case study with one of the reviewed game engines confirmed that the features of the game engine were actually the ones recorded in the literature and it was concluded that the features recorded in the study were valid for the other game engines as well. The game engine used in the case study was Unity 3D.

Another article with related content is “Open Source 3D Game Engines for Serious Games Modeling”. Navarro, Pradilla and Rios [3] focus on some aspects relevant to serious games and briefly describe the main features of six open source or free 3D game engines for serious games modelling. In conclusion, they propose the game engine JMonkey.

Trenholme and Smith [4] compare in a tabular form six game engines that are considered appropriate for developing first-person virtual environments. Besides recording the features of each game engine the authors highlight those features that each game engine is superior compared to the rest game engines.

Marks, Windsor and Wunsche [5] evaluate game engines in terms of their suitability for developing games for simulated surgical training. A list of available game engines is evaluated and three of them were selected as the most appropriate. The game engines selected were Unreal Engine 2, id Tech 4 and Source Engine. Based on a case study with these three game engines the authors concluded that Source Engine is the most appropriate for surgical training.

In most of the aforementioned studies the authors select their own list of criteria for analyzing and comparing a set of game engines of a specific type, such as 3D mobile game engines or open source 3D game engines for serious games modelling. There is no common framework for analyzing the game engines, making it difficult for the reader to realize if the features analyzed are just the most important ones, some or all the factors that should be considered when selecting a game engine for a specific purpose. Petridis et al. [6] propose such a game engine selection framework for high-fidelity serious applications, such as military systems simulations [7], and use it for evaluating four game engines focusing on serious games.

In our study we attempt to advance the state of the art on analyzing and comparing game engines by extending a proposed game engine selection framework for serious applications [6] and utilizing it for the comparative analysis of representative examples of game engines that fulfil different user needs. Specifically, we analyze: engines that do not require knowledge of programming; engines that are based on popular contemporary web technologies; engines that are

open source and can be customized/extended by experienced users; and professional game engines. All the selected game engines export games at least to Android and iOS mobile devices. In order to validate the results of the comparative analysis, a game is implemented in two of the selected game engines. Specifically, in our case study two similar shooter games are developed step-by-step according to tutorials provided by the official web sites of the two most powerful game engines included in our comparative analysis.

## 3. Game engines

In this section we briefly present some information for the game engines that were selected for the comparative analysis.

*GameMaker* was first released with the name Animo in 1991, and was suitable for creating 2D games. Although the game engine is suited for 2D games, it allows the user to add 3D graphics and physics. GameMaker is easy to use for *novice users*, since it does not require programming knowledge. However, the functionality of 3D camera compared to 3D graphics is limited [8]. The version we study in this paper is 2.0.1.8.

*JMonkey* was released for the first time in 2003. It is a free game engine and exports games at no cost to all platforms, including mobile devices. JMonkey supports 3D graphics. Although it is necessary to have experience in Java programming, it is an *open source engine* that enables users to expand it and adapt it to their needs [9]. The version we study in this paper is 3.1 beta 1.

The original version of *Marmalade* game engine was released in 2015. This engine was chosen as it exports games to many *mobile platforms* for free and provides 3D graphics rendering capabilities. In addition, Marmalade contributes to the creation of Desktop Web applications [10]. The version we study in this paper is 8.5.

The *OGRE 3D* game engine was first released in November 2013. Ogre3D is a free and open source engine. It allows exporting a game on iOS, Android and Windows Phone 8 and 3D graphics rendering [11]. The version we study in this paper is 2.1.

The *Shiva Game Engine* was released for the first time in July 2007. It provides 3D graphics rendering and has been used to develop very *popular games* such as the Prince of Persia 2 that was re-created for mobile devices using Shiva, and Babel Rising that was published by Ubisoft [12]. The version we study is 2.0.

The *Sio2* game engine was first released in 2009 and within a year it became one of the 3 most *popular* game engines used in the App Store [13]. Sio2 is a game engine that was previously free and open source; however, it is now available to users by payment. One can also access its source code, by purchasing the Certified Developer version. A trial version is provided to users with the most features available, such as mobile export. Still, game simulation is provided on the various iOS devices and is one of the most popular game engines to export to iOS platforms by providing advanced 3D graphics [14]. The version we study in this paper is 1.0.2.

The *Turbulenz* game engine was created in April 2013 and is the only engine we study that allows games to be exported to *mobile browsers*, based on the promising approach of web technologies. At the same time, Turbulenz is free, open source and supports 3D graphics rendering [15]. The version we study in this paper is 1.3.2.

The *Unity* Game engine was first publicly announced at Apple's Worldwide Developers Conference in 2005 [1] and is one of the best-known game engines. Unity provides advanced 3D graphics rendering and exports to mobile devices for free. Still, it is one of the most popular engines in the gaming industry, with which many successful games have been created, such as Deus Ex: The Fall, for mobile phones, or Assassin's Creed: Identity [16]. The version we study in this paper is 5.3.4.

*Unreal Engine*'s original release was in 1998, while Unreal Engine 4 was released for the first time in May 2012. This engine is suitable for rendering 3D graphics, is free and open source and is also one of the most popular game engines, as many games have been created with it and succeeded in the industry such as Absolver [17]. The version we study in this paper is 4.13.1.

## 4. Comparison framework and methodology

The selected game engines were analyzed using the framework presented in Figure 1. The framework is based on the "Game Engines Selection Framework for High-Fidelity Serious Applications" by Petridis et al. [6], supplemented with some additional categories of features


ISSN: 2384-8766 http://dx.doi.org/10.17083/ijsg.v4i4.194

(Development features and Deployment platforms) proposed by Pattrasitidecha [2]. Several features included in this framework are also analyzed in relevant literature and references are provided in the comparative analysis tables for interested readers.

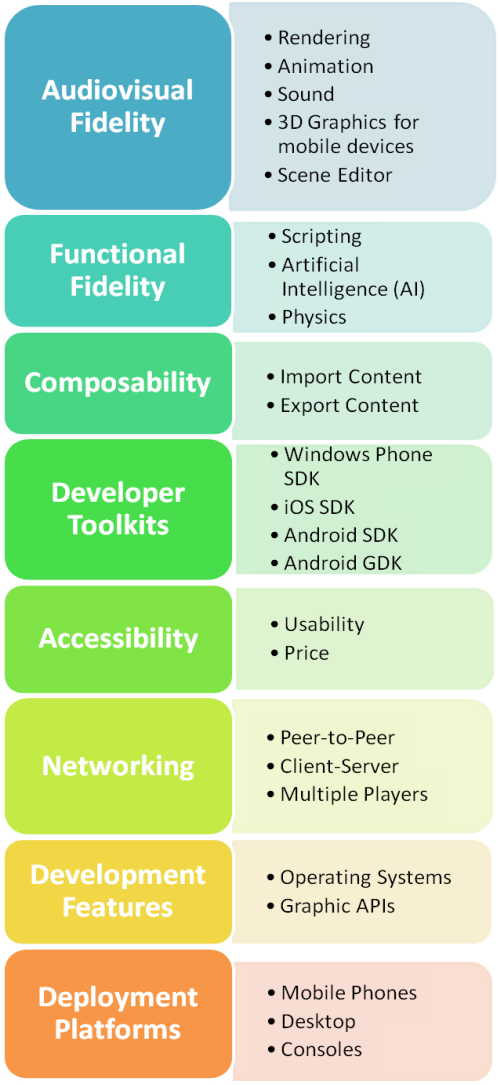


**Figure 1.** Comparison Framework based on Petridis et al. [6] & Pattrasitidecha [2]

*Audiovisual Fidelity* refers to how much the world of the game seems visually and acoustically realistic [6]. *Functional Fidelity* is defined by the degree to which the simulation operates correctly when game engines respond to user processes [7]. The concept of *Composability* is used to describe the reuse of content created within a game engine, but also the ability of the engine to import and use data from common or commercial sources [6]. *Developer toolkits* refer to SDKs and GDKs that developers have access in order to connect game engines with peripheral devices or other software APIs [6]. *User Accessibility* refers to the usability and price of a game engine and it is a very important criterion for selecting it as appropriate for ones purpose. *Networking* is crucial in some types of games, such as MMORPG, because multiple users have different characteristics that affect the nature of the game and are defined in its early stages of development [2]. The category of *Development Features* refers to the operating system on which a game engine can be used [2] and the Graphic APIs that the engine supports [18]. Finally, *Deployment Platforms*, or else *heterogeneity*, is a fundamental challenge for users of virtual environments design, as the ability to develop games on platforms for a wide range of hardware and software is a remarkable advantage [6].

The framework proposed by Petridis et al. [6] and adapted appropriately for our study, refers to selection criteria of game engines for the development of serious games, which have similarities but important differences as well from entertainment games. Although it might not be immediately

obvious, the categories of features comprising the framework represent important dimensions of serious games. For example, audiovisual and functional fidelity are important for achieving *immersion* and *flow* in serious games [6]. Networking is important for *multi-user* serious games, *interaction between learners and instructors* that take the form of virtual characters, or even serious games utilizing *social networking technologies* [6]. Networking provides also the chance to collect data during playing the game and carrying on learning activities, which are necessary for *user assessment*. Composability is important for serious games that aim to *model real world places and situations* [6] and require importing existing content from external sources. Moreover, composability promotes *modularity*, since it gives the chance to separate and restructure components of a given system [19] and reuse assets either in the same or other games [1]. Deployment platforms, or else heterogeneity, are crucial for every game, but especially for serious games. Supporting various deployment platforms combined with networking offer capabilities for *interoperability*, as well as the basis for offering *games as a service*. The ability to share resources in a heterogeneous environment is extremely important for serious games that have educational purposes and their target group uses a wide range of hardware and software platforms [6].

The comparative analysis of the selected game engines is carried out presenting the results for each category of features in separate tables. The information included in the tables was extracted from the official web sites of the game engines, user manuals and documentation. Moreover, information from relevant literature is provided in the tables using appropriate references. Red colour in tables 1 to 8 indicates the game engines which have a disadvantage compared to the others. On the other hand, green colour indicates the game engines which have an advantage compared to the others. We consider one game engine to be disadvantageous to the others, if half and more engines have the feature, while it does not. In addition, one engine may be considered to be disadvantaged by a sub-set of features (such as shadow rendering), if it has the fewer sub-group features in combination with the previous hypothesis. Similarly, we consider an engine to be advantageous to a feature or set of features in relation to the rest of the engines. The notation used in the tables is the following:

- "√": feature found in the engine's official web site and in some cases confirmed in the literature
- "x": feature not supported based on the official web site of the engine and in some cases confirmed by the literature
- "x?": feature not found neither in the information available to the official web site nor in the literature. However, we cannot be sure that this feature is not supported.
- In Table 4, the minus symbol (–) indicates that this feature in not present in the game engine, since it is not needed.
- In Table 5, besides the symbols √ and x, a scale of 1 to 5 (5 is the most positive evaluation) is used to assess the usability of game engines. *Unity*, *Marmalade*, *Sio2*, *Ogre3D* and *Shiva* game engines were evaluated by Pattrasitidecha's [2]. The remaining machines were evaluated in this article in a similar way based on examining tutorials and assets from their official web sites.

## 5. Results of the comparative analysis

### 5.1 Audiovisual fidelity

In Table 1 the results of the comparative analysis regarding the features falling into the category of audiovisual fidelity are presented. *Game Maker* is the least developed game engine in terms of audiovisual fidelity, since the only category in which it supports all the features is sound. *JMonkey* is one of the best game engines in the category of *static global illumination* together with *Unreal Engine 4* that is even better and *Ogre3D*, but it does not support lighting per pixel, audio streaming and occlusion culling. *Marmalade* also does not support lighting per pixel and ambient occlusion, anisotropic reflection, environmental mapping, lens flare and blend animation. *Ogre3D* has a benefit in the *shadows* category along with Unreal Engine. *Shiva* is an engine that has both an advantage and disadvantage in the features of *animation*, and specifically *forward animation* is a strong point in contrast with *morphing animation*. We notice that *Sio2* has a great disadvantage in *3D graphics for mobile devices* and *editor mapping* as it does not support any of the relevant features we've been examining in these categories. At the same time, it does not support 2D sound, morphing animation, blending animation, gloss/specular maps and procedural texture. *Turbulenz*

does not support light mapping, morphing animation, blends, 2D sound, and bump mapping. *Unity* does not support just multiple textures, while on the other hand it supports all the other features and seems to outperform the rest of the engines in animation along with *Unreal Engine 4*. *Unreal Engine 4* seems to outperform audiovisual fidelity as it supports all features and has an advantage in animation (along with *Unity* as mentioned), *shadows* and *global illumination*.

**Table 1.** Audiovisual fidelity.

| ***Features/ Game Engines*** | **Game Maker** | **JMonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1.Rendering**[4] | | | | | | | | | |
| 1.1 Texture[6] | | | | | | | | | |
| 1.1.1 Basic | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.1.2 Multi-textural | x | √ | √ | √ | √ | √ | √ | x[6] | √ |
| 1.1.3 Procedural | x | √ | √ | √ | √ | x? | √ | √ | √ |
| 1.2Lighting[6] | | | | | | | | | |
| 1.2.1 Per-Vertex | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.2.2 Per-Pixel | x | x[18] | x? | √ | √ | √ | √ | √ | √ |
| 1.2.3 Light Mapping | x | √ | √ | √ | √ | √ | x? | √ | √ |
| 1.2.4 Gloss/Specular Maps | x | √ | √ | x? | √ | x? | √ | √ (alpha channel) | √ |
| 1.2.5 Anisotropic Reflection | x | √[20] | x? | x? (normal reflection) | √[21] | √[22] | √[23] | √ | √ |
| 1.3 Shadows[6] | | | | | | | | | |
| 1.3.1 Shadows Volume | x | √[18] | x? | √ | x[24] | x? | x? | x[6] | √ |
| 1.3.2 Shadow Mapping | x | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.3.3 Projected | x | x[18] | x? | √ | √ | √ | x? | x[6] | √ |
| 1.3.4 Projected Planar | x | x[18] | x? | √ | x | x? | x? | √ | √ |
| 1.4 Special Effects[25,26] | | | | | | | | | |
| 1.4.1 Environmental Mapping | √ | √ | x? | √ | √ | √ | √ | √ | √ |
| 1.4.2 Particle System | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.4.3 Billoarding | √ (3D) | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.4.4 Lens Flare | x | √ | x? | √ | √ | √ | √ | √ | √ |
| **2. Animation**[6] | | | | | | | | | |
| 2.1 Forwards/ Kinematics[27] | x | x[18] | x? | x | √ | x | x? | √ | √ |
| 2.2 Inverse Kinematics[18] | x | x[18] | x? | √ | √ | x | x? | √ | √ |
| 2.3 KeyFrame Animation[28] | x | √ | √ | x | √ | √ | √ | √ | √ |
| 2.4 Skeletal Animation[29] | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 2.5 Morphing Animation[30] | √ (2D) | x[18] | √ | √ | x? | x | x? | √ | √ |
| 2.6 Blending[31] | x | √ | x? | √ | √ | x? | √ | √ | √ |
| **3. Sound**[1,31] | | | | | | | | | |
| 3.1 2D Sound[1,31] | √ | √ | √ | √ | √ | x | x? | √ | √ |
| 3.2 3D Sound[1,31] | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3.3 Streaming sound[31] | √ | x[18] | √ | √ | √ | √ | √ | √ | √ |
| **4. 3D Graphics for mobile devices**[2] | | | | | | | | | |
| 4.1 Mapping | | | | | | | | | |
| 4.1.1 Map Editor[2] | | | | | | | | | |
| 4.1.1.1 Bump Mapping[2] | x | √ | √ | √ | √ | x[2] | x? | √ | √ |
| 4.1.1.2 Parallax Mapping[2] | x | √ | x[2] | √ | √ | x[2] | x? | √ | √ |
| 4.1.1.3 Normal Mapping[2] | x | √ | √ | √ | √ | x[2] | √ | √ (not on mobile) | √ |
| 4.2 Global Illumination[2] | | | | | | | | | |
| 4.2.1 Ray tracing[2] | x? | x? | x? | √ | √ | x[2] | √ | √ | √ |
| 4.2.2 Ambient Occlusion[2] | x | √ | x[2] | √ | √ (not on mobile) | x[2] | √? (environmental lighting) | √ (not on mobile) | √ |
| 4.2.2.1 Cut Scene Editor[2] | x | √ | x[2] | x? | x[12] | x[2] | x? | √ (not free) | √ (Matinee) |
| **5. Scene Editor** | | | | | | | | | |
| 5.1 Occlusion Culling[18] | x | x[18] | x? | √ | √ | √ | √ (occlusion quarries) | √ | √ |

### *5.2 Functional fidelity*

The results regarding functional fidelity are summarized in Table 2. As we can observe, *Unity* and *Unreal Engine 4* both support all the features of functional fidelity and have the PhysX SDK for physical simulation which is suitable for this purpose [32]. Instead, *Turbulenz* displays the most deficiencies in Artificial Intelligence and Physics.

**Table 2.** Functional fidelity.

| ***Features/ Game Engines*** | **Game Maker** | **JMonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Scripting**[1] | | | | | | | | | |
| 1.1 Programming Language[2,3,6,26] | GameMaker Language | Java | C, C++ | C++ | Lua (+ Shiva) | C, C++, Lua | Javascript, HTML | C#, JavaScript, Boo | C++, Blueprint |
| 1.2 Programming Editor[2] | √ | √ | x[2] | √ | √ | x | √ (text editor) | √ (MonoDevelop) | √ (Kismet) |
| 1.3 GUI Editor[2] | x | √ | √ | √ | √ | x[2] | x | √ | √ |
| 1.4 Debugger[2] | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.5 Profiler[2] | √ | √ | iOS SDK | √ | x[2] | √ (not in free version) | √ | √ | √ |
| **2. Artificial Intelligence**[2,4,33] | | | | | | | | | |
| 2.1 AI Editor[2] | x | √ | x[2] | x[2] | √ | √ | x? | √ | √ |
| 2.2 Path Finding[2] | √ | √ | x | √ | √ | √ | x | √ | √ |
| 2.3 Decision Making[2] | x | √ | x | x? | √ | x? | x? | √ | √ |
| 2.4 Collision Detection[2] | √ (2D) | √ | x[2] | √ | √ | √ | √ | √ | √ |

| **3. Physics** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3.1 Physics Tools | | | | | | | | | |
| 3.1.1 Animation Editor[12] | √ | x? | √ | √ | √ | x[2] | x? | √ | √ |
| 3.1.2 Material Editor[2] | x | √ | √ | √ | √ | x[2] | x? | √ | √ |
| 3.1.3 Particle Effect Editor[2] | x | √ | x[2] | √ | √ | x[2] | x? | √ | √ |
| 3.1.4 3D Audio Editor[2] | √ | x? | √ | x[2] | √[2] | x[2] | x? | √ | √ |
| 3.2 Basic Physic[6] | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 3.3 Rigid Body[2] | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 3.4 Vehicle Dynamics[2] | x | √? | x? | x[18] | √ | √? | x? | √ | √ |
| 3.5 3D Physic SDK[2] | Bullet | Bullet | ODE | x[2] | ODE | Bullet | x? | PhysX | PhysX |

## 5.3 Composability

As shown in Table 3 all the engines, with the exception of *GameMaker* that is basically a 2D game engine, support importing/exporting of content from/to the best-known CAD platforms. *GameMaker* imports 3D animation models only from Misfit Model 3D software, which is old and without support.

**Table 3.** Composability.

| ***Features/ Game Engines*** | **Game Maker** | **Jmonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Import/Export content**[6,34,35] | | | | | | | | | |
| 1.1 MAYA | x | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.2 3D Studio MAX | x | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.3 Blender | x | √ | √ | √ | √ | √ | √ | √ | √ |

## 5.4 Developer Toolkits

In Table 4 the built-in SDKs and GDKs included in each engine are presented. *Unity*, *GameMaker* and *Sio2* require the separate installation of SDKs to export to iOS, Android or Windows Phone, while the other engines include the required SDKs automatically.

**Table 4**. Developer Toolkits.

| ***Features/ Game Engines*** | **Game Maker** | **JMonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Developer Toolkits**[6,36] | | | | | | | | | |
| 1.1 Windows Phone SDK | x | - | √ | √ | √ | - | - | √ | √ |
| 1.2 iOS SDK | √ | √ | √ | √ | √ | x | √ | x | √ |
| 1.3 Android SDK | x | √ | √ | √ | √ | x | √ | x | √ |
| 1.4 Android GDK | √ | √ | √ | √ | √ | x | √ | x | √ |

## 5.5 Accessibility

According to Table 5, *Unity* seems to be the most usable game engine, providing the most free tutorials, examples and assets, while its community is very large. However, technical support is not

offered in the free version, unlike the *Unreal Engine 4* and *Turbulenz* engines that provide free technical support. Moreover, the engines *JMonkey*, *Ogre3D*, *Turbulenz* and *Unreal Engine 4* are completely free and open source. The *Sio2* game engine, although previously provided free to users, today's all-in-one version with access to the engine's source code is priced at $ 1999, making it more expensive than any other.

**Table 5.** Accessibility.

| ***Features/ Game Engines*** | **Game Maker** | **Jmonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Usability** | | | | | | | | | |
| 1.1 Easy of learning | | | | | | | | | |
| 1.1.1. Tutorials[2] | 3 | 4 | 4[2] | 4[2] | 3[2] | 3[2] | 2 | 5[2] | 4 |
| 1.1.2. Examples[2] | 2 | 4 | 4[2] | 4[2] | 3[2] | 3[2] | 3 | 5[2] | 4 |
| 1.2 Docs and Support[2] | | | | | | | | | |
| 1.2.1. Docs Quality[6] | Docs and Tutorials in official | Docs and Tutorials in official | Docs and Tutorials in official site | Docs and Tutorials in official | Docs and Tutorials in official site | Docs and Tutorials in official site | Docs and Tutorials in official site, not | Docs and Tutorials in official | A subset of tutorials is at Unreal |
| 1.2.2. Technical Support[6,2] | x | x | √ (not in free version) | X | √ (Basic and Advance | √ (not in free version) | √ | √ (Professional | √ |
| 1.2.3. Community Support | 3 | 3 | 3[2] | 4[2] | 3 | 3[2] | 2 | 5[2] | 4 |
| **2. Price**[6,2] | | | | | | | | | |
| 2.1. Free | Studio Free | √ | Free version | √ | Web Edition | Trial version | √ | Personal version | √ |
| 2.2. Open Source | x | √ | x[2] | √ | x[2] | Certified Develope | √ | Professional | √ |
| 2.3. Price/Year | Studio Professional version: 149,99$+299,99$, | - | Indie version: 499$, Plus version: 1500$, | - | Basic new version: 200$ + tax, Advanced new version: | SIO2 Certified Developer version: 1999$, SIO2 Licensed | - | 395$: Plus version, 1500$ Professional version | 5% Commission after first 3000$ of the first game / 3 |

## 5.6 Networking

As shown in Table 6, all the game engines follow the same network model and can export games that refer to many users. *Turbulenz*, in addition to the client-server model, also uses the peer-to-peer model.

**Table 6.** Networking.

| ***Features/ Game Engines*** | **Game Maker** | **Jmonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Networking** [6] | | | | | | | | | |
| 1.1. Client-Server | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.2. Peer-to-Peer | x | x[18] | x | x[18] | x[2] | x | √ | x[6] | x[6] |
| 1.3. Multiplayer | √ | √ | √ | √ | √ | √ | √ | √ | √ |

### 5.7 Development features

As we can see in Table 7, *GameMaker*, *Marmalade* and *Unity* cannot be installed and run on Linux platforms, while the first cannot be installed on MacOSX as well. All game engines support both DirectX (or Direct3D) and OpenGL, except for *JMonkey*, *Marmalade* and *Sio2* which only support the OpenGL library. In addition, *Turbulenz* supports the Graphics API WebGL, which is suitable for graphics in browsers. *GameMaker* has the disadvantage that it runs only on Windows.

**Table 7.** Development features.

| ***Features/ Game Engines*** | **Game Maker** | **JMonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Operating Systems**[2,18] | | | | | | | | | |
| 1.1.Windows | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| 1.2. Mac OSX | x | √ | √ | √ | √ | √ | √ | √ | x |
| 1.3. Linux | x | √ | x[2] | √ | √ | √ | √ | x | x |
| **2. Graphics API**[18] | Direct X 9, OpenGL | OpenGL | OpenGL | Open GL, Direct 3D | Direct X 11, Open GL | OpenGL | WebGL | Direct X, OpenGL | Direct X 9, OpenGL |

### 5.8 Deployment platforms

From Table 8, we can conclude that *Marmalade* is the most suitable engine for mobile game development, while *JMonkey*, *Turbulenz* and *Unreal Engine 4* are not so strong in this feature. As for desktop platforms *JMonkey*, *Marmalade*, *Ogre3D* and *Sio2* do not support all available platforms. *Unity* has an important advantage in game development for consoles since it supports all the consoles we've been examining.

**Table 8.** Deployment platforms.

| ***Features/ Game Engines*** | **Game Maker** | **JMonkey** | **Marmalade** | **Ogre3D** | **Shiva** | **Sio2** | **Turbulenz** | **Unity** | **Unreal Engine 4** |
|---|---|---|---|---|---|---|---|---|---|
| **1. Mobile Devices** | | | | | | | | | |
| 1.1 iOS[2] | √ (not in free version) | √ | √ | √ | √ (not in free version) | √ | √ (browsers) | √ | √ |
| 1.2. Android[2] | √ (not in free version) | √ | √ | √ | √ (not in free version) | √ | √ (browsers) | √ | √ |
| 1.3. WebOS(Palm)[2] | x | x | x | x[2] | x | √ | x | √ | x |
| 1.4. Windows Mobile 6[2] | x | x | √ | x[2] | x | x[2] | x | x | x |
| 1.5. Windows Mobile 7[2] | √ (not in free version) | x | √ | x[2] | √ (not in free version) | x[2] | x | x | x |
| 1.6. Windows Phone 8 | √ (not in free version) | x | √ | √ | √ (not in free version) | x | x | √ | x |
| 1.7. Bada[2] | x | x | √ | x[2] | √ (not in free version) | √ | x | x | x |
| 1.8. Symbian[2] | x | x | √ | x[2] | x | x[2] | x | x | x |

| **2. Desktop** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2.1. Windows[2] | √ | √ | √ (not in free version) | √ | √ (not in free version) | √ | √ (browsers) | √ | √ |
| 2.2. Mac OSX[2] | √ | √ | √ (not in free version) | √ | √ (not in free version) | √ | √ (browsers) | √ | √ |
| 2.3. Linux[2] | √ | √ | x | √ | √ (not in free version) | x | √ (WebGl) | √ (Linux/ Stream OS) | √ |
| 2.4. Web[2] | HTML5 (not in free version) | x | Google Native Client (not in | x[2] | √ (with plug-in) | x[2] | √ | WebGL | HTML5 |
| **3. Consoles** | | | | | | | | | |
| 3.1. Playstation[2] | √ (not in free version) | x | x[2] | √ | √ (not in free version) | x[2] | x | √ | √ |
| 3.2. Wii[2] | x | x | x[2] | x[2] | √ (not in free version) | x[2] | x | √ | x |
| 3.3. XBOX[2] | √ (not in free version) | x | x[2] | √ | √ (not in free version) | x[2] | x | √ | √ |

## *5.9 Summary*

Based on the results of the comparative analysis we can draw some important conclusions regarding the effectiveness of each game engine for the categories of features investigated in this study:

It was noticed that *Game Maker* is not suitable for developing games with 3D graphics. However, it is ideal for creating games with 2D graphics easily, providing a user friendly environment and the ability to develop games without any experience in programming.

*JMonkey* falls short on some aspects of audiovisual fidelity, like the absence of occlusion culling or animation and 3D sound tools. Moreover, it cannot export games to consoles. On the other hand it is an open source game engine, enabling users to extend its functionality.

*Marmalade* is an engine that falls short on several features in each category, but it is an engine that has an advantage in the category of deployment platforms, as it can export games in most of the game platforms.

*Ogre3D*, like *JMonkey*, does not support many features of the audiovisual and functional fidelity category. On the other hand, *Ogre3D* is also an open source game engine and as a result it can be adapted to the needs of an experienced user. In addition, *Ogre3D* is ideal for shadow creation and suitable for beginners providing quite good support.

*Shiva* is not so accessible to users because it is not free and not also suitable for inexperienced users. However, it contains almost all the audiovisual features, while at the same time it outweighs in the physics subclass of functional fidelity, supporting all features.

*Sio2* is suitable for creating applications in iOS software, as its core is based on this operating system. It is deficient in many features of audiovisual and functional fidelity. It is also not so accessible to the users, because although it was previously free and open source, now it is the most expensive game engine.

*Turbulenz* is a game engine that falls short on several features under the functional fidelity category. However, it is open source and ideal for creating applications for browsers using the WebGL Graphic API.

*Unity* and *Unreal Engine 4* are the most powerful game engines, which support almost all of the features included in the framework used. *Unreal Engine 4* presents better audiovisual performance than *Unity*, as it supports more features while at the same time is an open source engine.

## 6. Empirical study

*Unity* and *Unreal Engine 4* have been selected by the review and benchmarking for further comparison, because they are clearly the two most developed game engines. In the context of the empirical study two similar desktop shooter games were developed on the two game engines through tutorials of their official websites, so as to make the best use of the various features of the two engines. Specifically, the Survival Shooter Game [37] was developed in *Unity* and the Twin Stick Shooter [38] in *Unreal Engine 4*. Then the games were processed to run on mobile devices and exported to be tested on an Android mobile device to get the results.

The features examined in the empirical study were the engine's usability, the learning curve, the process of exporting for mobile devices, as well as the game's quality and application size.

### 6.1 Usability

As far as the engine's usability and learning curve are concerned, *Unity* seems to be better in terms of a user-friendly interface and free assets for developing games, while it has fewer requirements for hardware. Specifically:

- *Unity* and *Unreal Engine 4* both integrate all the necessary tools for game development, but *Unreal Engine 4* has a more complex interface in comparison with *Unity* that offers all the necessary tools in a single window.
- In *Unity* the user must have knowledge (or learn) object-oriented programming and C# for programming the necessary scripts. On the other hand, the *Unreal* engine utilizes a Blueprint Visual Scripting system that is based on a node-based interface for defining objects and classes. This graphical editor can help a non programmer or a novice programmer, but for many programmers it might be preferable to write source code.
- Both engines have a wide community with many tutorials. In *Unity* most of the tutorials are in text format, while in *Unreal Engine 4* in video format. However, in *Unreal Engine 4* there are not so many free tutorials.
- Both engines have a wide asset store, but in *Unreal Engine 4* there are not many free assets in contrast with *Unity*.
- Finally, *Unity* has fewer requirements in hardware.

### 6.2 Exporting a game for mobile devices

The process of exporting the game to mobile phones is quite straightforward in both engines. Some notes are the following:

- In *Unity* normal mapping is actually not supported as was denoted in the comparative analysis of the game engines. However, there are many available shaders that can be used for avoiding errors. Another problem faced was a problematic joystick from the asset store. This problem was dealt with by writing code for programming the joystick using existing tutorials.
- In *Unreal Engine 4* there is no variety of mobile shaders and errors occurred. The shaders had to be modified using the material editor, while new settings for lighting had to be made. Moreover, there is no simulator module and the user has either to use a mobile previewer or launch the game while being developed. Finally, the installation is not straightforward and requires preparing and running the appropriate bat file using a computer.

### 6.3 Game quality and size

Using Game Profiles as well as monitoring the device's memory footprint during the game, we extracted the following information for the quality and the size of the final game. The device that we used had the following characteristics: Android 4.4, RAM 1GB, ROM 8GB, 1.2GHz Quad Core. The results are summarized in Tables 9 and 10.

**Table 9.** Game quality.

| | **Unity** | **Unreal Engine 4** |
|---|---|---|
| Problem with device memory | No | Yes |
| Memory footprint | Less than Unreal Engine | Four times more than Unity |
| **Profiler** | | |
| Draw Calls | 28 | 27 |
| Memory | Light 1.3 KB | Light 16 KB |
| Time | Animation in max time 2.43ms and avg<2ms | Animation in 8.03ms and avg<1.43ms |

**Table 10.** Application size.

| | **Unity** | **Unreal Engine 4** |
|---|---|---|
| Initial Textures size | 84.8 MB | 54.8 MB |
| Total game size after compression | 63.9 MB | 135 MB |
| Compression | Effective compression | Problem in game's compression |

In Table 10, we observe that in *Unity* the compression results in a game with a total size that is even smaller than the initial size of the textures used in the game. On the other hand, the compression of the game in *Unreal Engine 4* results in a game that has actually the same size of the corresponding project. It is obvious that game compression is much more effective in *Unity* and this is very important for games running in mobile devices.

### *6.4 User Experience*

User experience was quite different in the two games. In Table 11 a summary of user experience for both engines is presented.

**Table 11.** User experience.

| **Unity** | **Unreal Engine 4** |
|---|---|
| Movement with Joystick and shooting with touch | Movement and shooting through the Joysticks |
| Movement only with simultaneous shot | Ease of simultaneous movement and shooting |
| Targets more accurately | Not very accurate when shooting |
| Difficulty in simultaneous movement and shooting | Difficulty in using Joysticks sometimes |
| More realistic graphics | |

## *7. Conclusions and recommendations*

In conclusion, we cannot recommend a game engine as better than any other, since each engine has its advantages and disadvantages. The choice depends on the user's profile (e.g. pedagogists, domain experts, programmers) and knowledge, the result he wants to achieve, as well as his resources and time. This article advances the literature in the field by providing a holistic overview of game engines for (serious) games that cover different user profiles and needs. In contrast with existing comparative analyses and reviews of game engines that are based on specific features, such as providing the ability for 3D game development or being open source, our comparative analysis takes into account representative examples of game engines falling into different categories that cover a wide range of requirements. The comparative analysis is carried out using an extended list of important features recorded in a game engine selection framework for high-fidelity serious applications [6].

Specifically, the present article aims to:

- Support the user in choosing the right engine by setting a number of criteria to be taken into account.

- Present a comprehensive overview of a number of engines falling into different categories, such as engines that do not require knowledge of programming, engines that are based on popular contemporary web technologies, engines that are open source and can be customized/extended by experienced users and professional game engines.

The literature review and comparative analysis of game engines, based on the framework by Petridis et al. [6] and Pattrasitidecha [2], proved that the most dominant game engines are *Unity* and *Unreal Engine 4*. In the empirical study that followed, two simple desktop shooter games were developed in the aforementioned engines according to official tutorials provided and then they were adapted so as to run in Android devices. This empirical study confirmed the capabilities recorded for the two engines in the context of the comparative analysis of the nine engines investigated in this study. So, we can conclude that the information summarized for all the game engines is quite trustworthy. Of course in order to be sure the game engines have to be used in practice.

When it comes to the game engines tested in the context of the empirical study the most important conclusions can be summarized as follows:

- *Unity* was considered more suitable for beginners because: it has a simpler user interface; it provides many tutorials and examples; there are many available assets. Moreover, its installation does not require high-performance hardware. Regarding the development of mobile games for Android the process is quite straightforward and the export process is easier. However, C# programming is required.
- *Unreal Engine 4* is more suited to experienced users, as: it supports Visual Scripting and has a more complex graphical environment; it has a steeper learning curve. This engine requires high-performance hardware, but on the other hand its graphics are remarkable. The *Unreal Engine 4* is suitable for desktop games and provides more features for graphics rendering. However, bugs were detected in the process of transferring the game to Android, while difficulties were confronted in compressing the game, resulting in performance problems with devices having little memory available.

It is clear that this study has some limitations. First of all, the comparative analysis included a small number of game engines in comparison with the large number of engines that are available. However, we made an effort to include representative examples of well-known game engines falling under different categories, such as game engines for novice and experienced users, game engines based on web technologies, open source game engines that can be customized, game engines used in the games industry and game engines targeted mainly to mobile devices. The common feature of the game engines analyzed is that they support exporting a game at least to the two most used mobile platforms, namely Android and iOS. This comparative analysis of various game engines based on an extensive list or features can assist anyone interested in the field in drawing a clear picture on the various categories of existing game engines and also in making an informed choice based on his knowledge, the nature of the game to be developed and the most important game features that must be supported by the selected game engine.

Another limitation lies in the fact that the comparative analysis of game engines was based on information from the official game engines web sites and the available literature. A great effort was made to cross-check the information presented from various sources. Moreover, the empirical study with the two game engines confirmed the information presented in the context of the comparative analysis and this is a good indication that the review is valid.

The aforementioned limitations provide directions for future research in the field. The literature review and comparative analysis can be extended in order to include more game engines, which could also be investigated by utilizing them for developing prototype games. In addition, games can be exported to other platforms other than Android in order to explore more features of different engines.

What would be even more important is to develop a web application for presenting all this information to people interested in utilizing game engines. This application could even make proposals to users regarding the best choices of game engines for a specific project based on information provided by users in a specialized form regarding their knowledge on programming, their preferences on developing a game (e.g. coding, visual scripting, defining actions through a drag and drop interface and so on), the features that are considered important for their game, development and deployment platforms.

## References


[1] Andrade, A., Game engines: a survey, EAI Endorsed Transactions on Serious Games, 15(6): e8, 2015.

[2] Pattrasitidecha, A., "Comparison and evaluation of 3D mobile game engines", Master of Science Thesis in the Program International Design, Department of Computer Science and Engineering, University of Gothenburg, 2014.

[3] Navarro, A., Pradilla J. V. and Rios O., "Open Source 3D Game Engines for Serious Games Modeling". [ebook] Modeling and Simulation in Engineering, InTech, 2012. Available at: <http://www.intechopen.com/books/modeling-and-simulation-inengineering/open-source-3d-game-engines-for-serious-games-modeling> [Accessed 5 July 2016].

[4] Trenholme, D. and Smith, S. P., "Computer game engines for developing first-person virtual environments", Virtual Reality 12:181–187, Durham University, 2008.

[5] Marks, S., Windsor, J. and Wunsche, B., "Evaluation of Game Engines for Simulated Surgical Training", GRAPHITE, Perth, Western Australia, December 1–4, 2007. https://doi.org/10.1145/1321261.1321311

[6] Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M. and Freitas, S., Game Engines Selection Framework for High-Fidelity Serious Applications, International Journal of Interactive Worlds, Vol 2012 (2012), 19 pages.

[7] Alexander, A.L., Brunyé, T., Sidman, J. and Weil, S.A., "From Gaming to Training: A Review of Studies on Fidelity, Immersion, Presence, and Buy-in and Their Effects on Transfer in PC-Based Simulations and Games", Aptima, Inc., Woburn, MA, DARWARS Training Impact Group, 2012.

[8] "GameMaker," https://www.yoyogames.com/

[9] "JMonkey," https://jmonkeyengine.org/

[10] "Marmalade," https://www.madewithmarmalade.com/

[11] "OGRE 3D," http://www.ogre3d.org/

[12] "Shiva," http://www.shiva-engine.com/

[13] Mullen T., "3D for iPhone Apps with Blender and Sio2", Wiley Publishing, Inc, Indianapolis, Indiana and Canada, 2010.

[14] "SIO2," http://www.sio2interactive.com/index.php

[15] "Turbulenz," http://biz.turbulenz.com/home

[16] "Unity3d," https://unity3d.com/

[17] "Unreal Engine," https://www.unrealengine.com/

[18] Rocha, R. V., Rocha, R. V. and Araújo, R., "Selecting the best open source 3D games engines", In Proceedings of the brazilian Symposium on Games and Digital Entertainment, Florianópolis, Santa Catarina, Brazil, 2010.

[19] González, A., España, S. and Pastor, Ó., "Unity Criteria for Business Process Modelling: A theoretical argumentation for a Software Engineering recurrent problem". Third International Conference on Research Challenges in Information Science (RCIS), Fez, Morocco, April 22-24, 2009.

[20] "Global Anisotropic Filtering- JMonkey," https://hub.jmonkeyengine.org/t/global-anisotropic-filtering/27054

[21] "Shiva 3d engine," https://kuliahvrits.wordpress.com/2010/06/15/shiva-3d-engine/

[22] "Sio2- Project Summary," https://www.openhub.net/p/sio2

[23] "Making the Move to HTML5, Part 2," "http://www.gamasutra.com/view/feature/187014/making_the_move_to_html5_part_2.php?print=1

[24] "Shadow Optimization and Physics- Shiva," http://www.shivaengine.com/developer/forum/viewtopic.php?t=7051&p=7118

[25] Friedrich, H., Günther, J., Dietrich, A., Scherbaum, M., Seidel, H. and Slusallek, P.,. "Exploring the Use of Ray Tracing for Future Games", Sandbox Symposium 2006, Boston, Massachusetts, July 2006. https://doi.org/10.1145/1183316.1183323

[26] Fritsch, D. and Kada, M., "Visualization Using Game Engines", Institute for Photogrammetry (ifp), University of Stuttgart, Germany, 2004.

[27] Park, J. and Fussell, D.S., Forward dynamics based realistic animation of rigid bodies, Computers & Graphics, Volume 21, Issue 4, July–August, 1997. https://doi.org/10.1016/S0097-8493(97)00024-1

[28] Thiebaux, M., Marsella, S., Marshall, AN. and Kallmann, M., "Smartbody: Behavior realization for embodied conversational agents", Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems, May,12-16., Estoril,Portugal, 2008.
[29] Stokes, M., "Improving the Performance of Skeletal Mesh Animations in the Blender Game Engine", Master of Science in Computer Science, Eastern Washington University, Cheney, Washington, 2014.
[30] Jung, Y. and Behr†, J., "Extending H-Anim and X3D for Advanced Animation Control", Web3D '08 Proceedings of the 13th international symposium on 3D web technology, pp 57-65, New York, NY, USA, 2008.
[31] Basten, B.J.H. and Egges, A., "Evaluating distance metrics for animation blending", ICFDG April 26-30, Orlando, FL, USA.Bijl, J. L. and Boer, C. A. (2011, December). Advanced 3D visualization for simulation using game technology. In Proceedings of the Winter Simulation Conference (pp. 2815-2826). Winter Simulation Conference, 2009.
[32] Harris, A. C., "Design and implementation of an autonomous robotics simulator", Doctoral dissertation, The University of North Carolina at Charlotte, 2011.
[33] Maciel, A., Halic, T., Lu, Z., Nedel, LP., and De, S., Using the PhysX engine for physics-based virtual surgery with force feedback, The International Journal of Medical Robotics and Computer Assisted Surgery, Volume 5, Issue 3, pp. 341–353, September 2009. https://doi.org/10.1002/rcs.266
[34] Kosmadoudi, Z., Lim, T., Ritchie, S., Liu, Y., and Sung, R., Engineering design using game-enhanced CAD: The potential to augment the user experience with game elements, Computer-Aided Design, Volume 45, Issue 3, Pages 777–795, 2013. https://doi.org/10.1016/j.cad.2012.08.001
[35] Kosmadoudi, Z., Lim, T., Ritchie, J.M., Sung, R.C.W., Liu, Y., Stănescu I.A. and Ştefan A., "Game Interactivity in CAD as Productive Systems", Procedia Computer Science Volume 15, 2012, pp. 285–288, 4th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES'12), 2012. https://doi.org/10.1016/j.procs.2012.10.081
[36] Scacchi, W., "Computer game mods, modders, modding, and the mod scene", Peer Review Journal on the Internet, Volume 15, Number 5 - 3 May, 2010.
[37] "Survival Shooter Tutorial- Unity 3D," https://unity3d.com/learn/tutorials/projects/survival-shooter-tutorial
[38] "Blueprint Twin Stick Shooter- Unreal Engine 4," https://docs.unrealengine.com/latest/INT/Videos/PLZlv_N0_O1gb5sdygbSiEU7hb0eomNLdq/