Global Journal of Computer Sciences: Theory and Research

Volume 8, Issue 3, (2018) 126-131

www.gjcs.eu

Basic elements and characteristics of game engine

Ramiz Salama*, Near East University, Department of Computer Engineering, Nicosia 99138, Cyprus
Mohamed ElSayed, Near East University, Department of Computer Engineering, Nicosia 99138, Cyprus

Suggested Citation:
Salama, R. & ElSayed, M. (2018). Basic elements and characteristics of game engine. *Global Journal of Computer Sciences: Theory and Research*. *8*(3), 126–131.

Received from March 11, 2018; revised from July 15, 2018; accepted from November 13, 2018.
Selection and peer review under responsibility of Prof. Dr. Dogan Ibrahim, Near East University, Cyprus.
©2018 SciencePark Research, Organization & Counseling. All rights reserved.

Abstract

Contemporary game engines are invaluable tools for game development. There are many engines available, each of them which excel in certain features. Game Engines is a continuous series that helps us to make and design beautiful games in the simplest and least resource way. Game drives support a wide variety of play platforms that can translate the game into a game that can be played on different platforms such as PlayStation, PC, Xbox, Android, IOS, Nintendo and others. There is a wide range of game engines that suit every programmer and designed to work on Unity Game Engine, Unreal Game Engine and Construct Game Engine. In the research paper, we discuss the basic elements of the game engine and how to make the most useful option among Game Engines depending on your different needs and needs of your game.

Keywords: Game engine, game engine element, basics of game engine.

* ADDRESS FOR CORRESPONDENCE: **Ramiz Salama,** Near East University, Department of Computer Engineering, Nicosia 99138, Cyprus. *E-mail address:* ramiz.salama@neu.edu.tr / Tel.: +90 (392) 680 20 00

1. Introduction

Game Engine is hard to define, the term is somehow known in the gaming community but few who really know what it is. It is because game engine comes in many shapes, and the real time of making one wouldn't be done if you don't have experience in software development. Let's dug in a little more in-depth describing how a computer executes functions. A computer in its simplest form has a group of circuits with just on and off, after that the values combine together into functions eventually shaping an operating system, the operating system can then make and execute applications created by mid-high level programming languages, these languages are used to run game engines and other web browsers. Now we see 'computers work on a principle which can be described as layers of abstracted complexity' (Michaelenger). Game engines lie on a top of the ladder of complexity I mentioned, so the Wikipedia definition seems somehow efficient; '...a software framework designed for the creation and development of making video games'.

2. Research method

This research is an overview study in which the final result is a wide view on the basic elements of game engines.

3. Literature review

The aims of game engines:

3.1. Graphics

Let's say that we want to create a character, item or icon (referred to as a sprite) (Gregory, 2014) onto the main scene in our game. If we have to do this without the help of a game engine, the just simple function of displaying an image in the desired location could take 100 lines of code or much more but with an engine like Cocos2D (used for making iPhone games) it only takes this:

CCSprite *sprite = [CCSpritespriteWithFile:@"sprite.png"]; sprite.position = cpp(100, 100); [game addChild: sprite].

This is a good example of how game engines work in the back, actions that have already been programmed and implemented before by other programmers. A game engine can have many graphic features; most include functions for animations, lighting systems, fades, layers, scaling and resolution. All of these would take a much more of complex code; the language of the engine can be entered and executed more quickly (Gregory, 2014).

3.2. Physics

Many modern games use some shape of physics to react to real world actions between the world and the player. Most of us know Angry Birds; it uses physics to run its entire gameplay theory with falling, swinging and destructible blocks. Physics can mimic gravity, friction, velocity, bounciness, mass and other properties. Some popular physics engines are Box2d, Havok and PhysX and they are internally existed into different game engines. These physics code samples are very useful to developers because they contain complex mathematical equations that aren't simple to make from scratch (Gregory, 2014).

3.3. User input

It's the most important part of the game, the interaction of a game runs when the user presses a command and magic starts happening. Inputs can alter greatly depending on the type of the game

that is done and what platform it is for; mobile games run with touch commands, pc games likely run with the keyboard and the mouse, and console systems like PlayStation or Xbox use their own different controller. A game engine can control the process of making a game for one or many of the platforms in an advantage named input handling. The engine will pay attention for inputs, meaning that it is ready for a known key or control to be pressed, and when a press and/or release is identified, it will run an 'event'. The event can be anything from moving the player, character action or opening a menu (Gregory, 2014).

3.4. Scripts/behaviours

A script into a game engine is a chunk of code that can command when a reaction or event is supposed to happen, they are known also as behaviours in some engines because they dictate when and how things should do in the game. A script can tick an event like a new scene or level change, a changing in the lighting or camera, or a character, enemy or item being added or removed from the scene. These scripts, like many other advantages of a game engine, are pre-built sections of code that would otherwise be very long and time wasting as well as hard to write (Michaelenger, 2013).

3.5. GUI

GUI is an abbreviation that is known as a graphical user interface, it indicates to on-screen text and menus, information texts and instruction texts. Drawing text with certain fonts and colours on screen with interactive buttons isn't that hard to code though if a game engine contains a customisable GUI system, it can keep a huge amount of time from being lost. Every GUI is going to be superb and customisable, designing one that goes along the game's visual style and meeting its gameplay is very important. Below is a live example of how to add an interactive button with text into the Unity game engine which uses JavaScript as its main language (Michaelenger, 2013).

```
//JavaScript
function OnGUI (){if (GUI.Button (Rect (10,10,150,100), "I am a button")){ print ("You clicked the button!"); }}
//C#
using UnityEngine;
using System.Collections;
public class GUITest : MonoBehaviour{void OnGUI () {
if (GUI.Button (new Rect (10,10,150,100), "I am a button"))
{print ("You clicked the button!");}}}
```

3.6. Sound

Engines can process sound and music. Game engines can help developers by playing sounds and music within specific events, they can also put the audio in a 3D field which will make the sounds run from a particular area alongside the games environment. The sound features in an engine can change volume, panning, distance and also add pitch modulation depending on the features of the environment. These ways of changing and handling audio give a game depth and create a wonderful experience. Most of the time audios get neglected within the game, but if the audio were of low quality the players would notice and make the game sink to failure (Coleman & Jahmel).

3.7. Porting

An extra big advantage of using a game engine, and one of the reasons that most developers will prefer to use one rather than building from the start, is that engines typically provide a means to publish/port to multiple platforms (Android, IOs, Web, PS, Xbox, PC, Steam, etc.) (Nelson, 2017).

3.8. Networking

Many modern games use some shape of online play components and it can be a great selling point or even a key element the gameplay moves around. Engine-provided scripts can help resolve communication problems between servers and devices that arise when offering online gaming features (Unity3d.com).

3.9. Types of game engines

Game engines come in many categories that cater to different types of games, developer preferences, team and project sizes and levels of programming skills. The differences between engines, how they are categorised and used are vast and even arduous, there are literally thousands of game engines (Coleman & Jahmel).

3.10. Mobile, 2D, and 3D

Some of the game's engines meet the special needs of mobile games. They contain scripts to transfer to Android, HTML5, iOS and improved controls for touch screens. Many 2D gaming engines allow you to publish on mobile, while others allow you to port to consoles as well. 3D game engines are more complex and multifaceted than 2D. 3D worlds in engines such as Unreal and Unity include huge terrain, models, lighting and other aspects that place them in a completely different category (Instabug.com, 2018; Ward, 2008).

3.11. API's, SDK's and visual interfaces

API stands for the API, it uses a specific type of programming language to create a library for coding functions. SDK is a software development kit, which combines several APIs together and a variety of development tools such as modelling, animation, environments and physics. Examples include Adobe Gaming SDK, Corona SDK and Valve's Source SDK. Most SDKs are supported in online communities with examples of coding examples of the functions and features they contain. Most of these APIs or SDKs are not visible, there are no navigation buttons or menus, and users are required to start from the script/code to get and run a game, and much programming knowledge is required. Recently there is a tendency to provide a visible interface to the engine, it converts the engine coding into a complete software tool, which can be used by someone who is not a programmer. Some visual editors such as the GameSalad, Stencyl, Gamemaker and 3D Engine Unity 2D engines are excellent at developing high-speed creative development, attracting independent developers or small teams. Another major twist in these visual editors is their perception of game scripts (Kalinin, 2016). An example of the visual code in the Stencyl engine, where code generation is provided in colour-coordinated blocks of logic that are grouped together with a tab right next to it where the actual text can be displayed. While the visual editor's engines are very elegant and streamlined, they are said to be very limited at times. The more the engine helps to build by reducing the necessary coding, the less control is usually available to the developers to do exactly what they think of the game (Kalinin, 2016).

3.12. Monetisation

One category relates to how the engine is available for use and the engine is either open source, free, commercially/owned and sometimes not available abroad to use the company that created it. Open source engines are available for any use by the public, with source code that can be changed by anyone. Open source engines are often cooperative efforts. Freeware engines are distributed over the Internet to anyone who uses them, but the source code is not necessarily available to everyone (Hunter, 2017).

Commercial game engines are marketed either to large companies or small teams that will purchase different types of coding libraries/engines and meet their project. Commercial engines are converted into money in a variety of ways and this is where the complexity is somewhat complicated. The engine can be sold at a price for any use (such as TorqueGameBuilder or Gamemaker). It can also be a subscription-based license (such as Unreal) and other times the engine will be established on royalty, which means developers will owe the engine creators a certain percentage of their profits once they start The game is in making money (Like Unity3d) (Hunter, 2017).

4. Results

In game development in the past, teams and companies required that they have programmers who build everything from the beginning. As time advanced, the game industry has evolved and their accumulated knowledge has begun to overcome. Game makers are no longer exposed to complexities that take a long time because many technical aspects have been solved by those that preceded them. Game programming is in a collaborative state, it grows on itself dramatically and the output of this is game engines.

Developers no longer need to start from zero. This is why gaming has been able to progress quickly. Game engines have conquered the way for greater creativity, diversity and pure amazement in games. Speed development of engines put developing game making tools in the hands of indie teams and amateurs, promoting education, help developing the industry to new and exciting forms.

5. Conclusion

Finally, to make a decision you have to decide which platform you will port your game to and how to market it away in a real-time market.

However, now that new consoles and devices are being made, Unity seems to support at most every new product. This is an extremely big advantage since using Unity allows you to port to platforms that other engines don't support. Even though learning Unity is hard but the reward is much more worth it.

Unreal Engine seems to be a better option than Unity for games which need graphically intensive performance and for companies that have very experienced development teams. However, for small companies looking to target multiple platforms and market effectively, Unity wins the battle.

References

Coleman & Jahmel. *'Game engines (types and purposes)'*. Wordpress.com. Retrieved from https://jahmelcoleman.wordpress.com/games-development/game-engines/

Gregory, J. (2014). *Game engine architecture*, (2nd Ed.), Florida, USA.

Hunter, C. (2017). *'10 Monetization Tips for Mobile Games in 2017'*. Retrieved from https://gameanalytics.com/blog/monetization-tips-mobile-games-2017.html

Instabug.com. (2018). *'Top game engines in 2018'*. Retrieved from https://blog.instabug.com/2017/12/game-engines /

Kalinin, V. (2016). *'Android platforms: what game engines, libraries, and APIs should I choose?'*. Retrieved October 26, 2016 from https://software.intel.com/en-us/articles/android-platform-what-game-engines-libraries-and-apis-should-i-choose

Michaelenger (2013). *'Game engines: how do they work?'* Giantbomb.com. Retrieved June 20, 2013 from https://www.giantbomb.com/profile/michaelenger/blog/

Nelson, X. Jr. (2017). *'Why porting games to PC is hard?'* pcgamer.com. Retrieved October 5, 2017 from https:// www.pcgamer.com/why-porting-games-to-pc-is-hard/

Unity3d.com. *'Game engines-how do they work?'*. Retrieved from https://unity3d.com/what-is-a-game-engine

Ward, J. (2008). *'What is a Game Engine?'*. Retrieved April 29 2008 from http://www.gamecareerguide.com/features/529/what_is_a_game.php?page=1