CHALMERS


| Features | UDK | Unity3D iOS Pro | Shiva3D Advance | Marmalade | Dx Studio | Esenthel Engine | SIO2 | Ogre3D | Unigine | SunBurn | Raydium | oolongengine | jPCT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1. Programming Libraries** | | | | | | | | | | | | | |
| 1.1 3D Graphics | | | | | | | | | | | | | |
| 1.1.1 Bump Mapping | o | o | o | o | x | o | x | o | o | x | x | x | x |
| 1.1.2 Normal Mapping | o | o | o | o | o | x | x | o | o | o | x | x | x |
| 1.1.3 Parallax Mapping | o | o | x | x | o | x | x | x | o | x | x | x | x |
| 1.1.4 Static Global Illumination | o | o | x | x | x | x | x | x | o | x | x | x | x |
| 1.1.5 Ambient Occlusion | o | Not on mobile | Not on mobile | x | x | Not on mobile | x | x | o | o | x | x | x |
| 1.2 3D Physics | PhysX | PhysX | ODE | ODE | PhysX & Bullet | PhysX & Bullet | Bullet | x | o | o | ODE | Bullet | o |
| 1.3 Artificial Intelligence | o | x | x | x | o | o | o | x | o | x | x | x | x |
| 1.4 Network | o | o | o | o | MMO | MMO | o | x | o | x | o | o | x |
| 1.5 Particle Effects | o | o | o | x | o | o | o | o | o | x | o | o | x |
| 1.6 3D Audio | fmod | fmod | o | o | o | o | OpenAL | x | o | o | o | o | x |
| 1.7 3D Video | Bink | o | o | o | Theora | x | o | x | o | x | o | x | x |
| **2. Usability** | | | | | | | | | | | | | |
| 2.1 Ease of Learning | | | | | | | | | | | | | |
| 2.1.1 Tutorials | 4 | 5 | 3 | 4 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 2 |
| 2.1.2 Examples | 4 | 5 | 3 | 4 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 1 |
| 2.1.3 Communities | 4 | 5 | 2 | 3 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 3 |
| 2.1.4 Programming Language | Unreal Script | C#, Javascript | LUA | C++ | Javascript | C++ | C++ | C++ | C++ | C# | C | C++ | JAVA |
| 2.1.5 Technical Support | x | x | x | x | o | o | o | x | o | x | x | x | x |
| 2.2 Efficiency of Use | | | | | | | | | | | | | |
| 2.2.1 Tools | | | | | | | | | | | | | |
| 2.2.1.1 Map Editor | o | o | o | o | o | o | Blender | x | o | o | x | x | x |
| 2.2.1.2 Animation Editor | o | o | o | o | o | o | Blender | o | o | x | x | x | x |
| 2.2.1.3 Material Editor | o | o | o | o | o | o | x | x | o | o | x | x | x |
| 2.2.1.4 AI Editor | o | x | x | x | o | o | x | x | x | x | x | x | x |
| 2.2.1.5 GUI Editor | Flash | x | o | o | o | x | x | o | x | x | x | x | x |
| 2.2.1.6 Particle Effects Editor | o | o | o | x | o | x | x | o | o | x | x | x | x |
| 2.2.1.7 Cut Scene Editor | o | Animation Editor | x | x | x | x | x | x | x | x | x | x | x |
| 2.2.1.8 3D Audio Editor | o | o | o | o | o | o | o | x | o | o | x | x | x |
| 2.2.1.9 Programming Editor | Visual Studio, Kismet | Mono Develop | o | o | o | Visual Studio | iOS SDK | iOS SDK | o | o | iOS SDK | iOS SDK | o |
| 2.2.1.10 Debugger | Visual Studio | Mono Develop | x | o | o | Visual Studio | iOS SDK | iOS SDK | o | x | iOS SDK | iOS SDK | o |
| 2.2.1.11 Profiler | x | o | x | iOS SDK | x | x | iOS SDK | iOS SDK | x | x | iOS SDK | iOS SDK | x |
| 2.2.2 Supported 3D Modeling Tools | | | | | | | | | | | | | |
| 2.2.2.1 MAYA | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.2 3D Studio MAX | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.3 Blender | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.4 XSI | o | o | o | o | o | o | o | o | o | o | o | o | o |
| **3. Development Platform** | | | | | | | | | | | | | |
| Windows | o | o | o | o | o | o | o | x | o | o | x | x | o |
| Mac OSX | x | o | o | o | x | o | o | o | o | x | o | o | o |
| Linux | x | x | x | x | x | x | x | x | o | x | x | x | o |
| **4. Deployment Platform** | | | | | | | | | | | | | |
| Mobile Phone | | | | | | | | | | | | | |
| -iOS | o | o | o | o | x | o | o | o | o | x | o | o | x |
| -Android | x | o | o | o | o | o | o | x | Some GPU supported | x | x | x | o |
| -WebOS(Palm) | x | x | o | o | x | x | o | x | x | x | x | x | x |
| -Windows Mobile 6 | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| -Windows Phone 7 | x | x | x | x | x | x | x | x | x | o | x | x | x |
| -Bada | x | x | AirPlay | o | x | x | o | x | x | x | x | x | x |
| -Simbian | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| Mobile Console | | | | | | | | | | | | | |
| -PSP | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| -NDS | x | x | x | x | x | x | x | x | x | x | x | x | x |
| Desktop | | | | | | | | | | | | | |
| -Web with plugin | x | o | o | x | o | x | x | x | x | x | x | x | x |
| -Windows | o | o | o | x | o | o | x | o | o | x | o | x | x |
| -Mac OSX | x | o | o | x | x | o | o | o | o | x | o | x | x |
| -Linux | x | x | x | x | x | x | x | x | x | x | o | x | x |
| Console | | | | | | | | | | | | | |
| -PS3 | x | o | x | x | x | x | x | x | o | x | x | x | x |
| -Wii | x | x | o | x | x | x | x | x | x | x | x | x | x |
| -XBOX | x | o | x | x | x | x | x | x | x | x | x | x | x |
| **5. Price** | | | | | | | | | | | | | |
| Price | $99/seat + 25% royalty after $50000 of income | $ 3000 for iOS Pro | 1499 Euro | $2500 per seat per year | $ 500 | $500 per 1 application | 399.99 $ for iOS and 399.99 $ for Android | Open source | $ 4950 | $ 300 | Open source | Open source | Open source |

# Comparison and evaluation of 3D mobile game engines

*Master of Science Thesis in the Programme Interaction Design*

*AKEKARAT PATTRASITIDECHA*

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, February 2014

Comparison and evaluation of 3D mobile game engines

Akekarat Pattrasitidecha



Examiner: Ulf Assarsson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden February 2014

**Abstract**
Game engines are vital for developing 3D applications and games today. This is because the engines significantly reduce resources---time and manpower---to implement mandatory but complex functionalities. However, with over 100 engines available with highly different ranges of features, performance, license, and cost, selecting an appropriate game engine for a specific objective becomes a challenging problem.

This thesis aims to mitigate this problem by developing a comparison matrix for selecting a game engine. We define criteria for selecting engines and critical aspects of mobile 3D game development to be included in the matrix. Then, we evaluate this matrix by a case study that is, from given requirements, selecting an engine by using the matrix.

Evaluation of the comparison matrix is carried out through case study, that advocate the correctness, and robustness of the comparison matrix.

**Keywords**
Game engine, mobile, 3D, comparison matrix

# Table of content

# 1 Introduction

This thesis focuses on creating a comparison matrix, which will help users pick the optimal 3D game engine on mobile platforms for their purposes. The thesis evaluates appropriate comparison criteria with a subset of chosen 3D mobile game engines. Then, all of the evaluation results will be showed in the comparison matrix.

## 1.1 Motivation

Nowadays, games are more popular to consumers than the past. Vgsales states that game industry is growing very fast. The game industry is larger than the movie industry (Arstechnica, 2007). Since many companies make a high profit from games, many developers create games.

In order to create a 3D game, implementers need to have various high-level skills because there are plenty of techniques within 3D games (Eberly, 2006). The magnitude of techniques used in 3D games for rendering and special effects is high (Eberly, 2006) . A modern competitive rendering engine for a graphics intensive game can require several man-years (Epic Games, 2011). A real-time application like a 3D game also requires numerous optimizations. But, plentiful 3D games are using the same optimizations and same set of core techniques. Therefore, many companies offer a game engine, which is system that provides core functionality of a real-time 3D game.

3D game engines help game companies reduce their cost, time, and resources, since a game producer can use ready-made functionalities of the engine. But, there are many game engines with different features. It is hard to choose the engine for different purposes. Once company chooses a game engine, their game features are limited to that engine. Cost of changing engine is extremely high. People who want to select a 3D game engine need to understand true functionalities of each engine, but supplier of the engines try to show their engines' advantages but hide their engines' disadvantages. Therefore, a comparison matrix from a rigorous study will help them to choose their optimal 3D game engine much easier.

## 1.2 Problem Area

This thesis focuses on developing a comparison matrix for 3D game engines on mobile platform. Since game frameworks have countless functionalities, the most important features should be included in comparison criteria. Taxonomy of each evaluation criterion should also be decided. Additionally, taxonomy should not be very complicated but provide enough information. Secondly, choosing a 3D game engine from all game engines is another problem because some game engines are not worth to compare. For example, game engines that lack important functionalities should be excluded. Third, there is no document that provides all

evaluation information for a specific engine. We need to investigate the entire set of selected 3D game engines ourselves (see Chapter 4).

## 1.3 Task and Objectives

Objective of this thesis is to create a matrix for comparing 3D game engines in mobile platform. Companies can use the comparison matrix to choose 3D game engine easier because important information of standout game engines is investigated. The comparison matrix should be easy to read and has enough information. In order to create comparison matrix, we decide to do these following tasks.

1. Comparison criteria selection: There are countless functionalities within game engines, but these are immoderate information for game engine evaluator. In order to compress information, the most important 3D game engine features should be decided. Comparison criteria are rows of comparison matrix.
2. Comparison criteria taxonomy determination: This step chooses appropriate taxonomy for each criterion. Taxonomy is shown directly to the reader, so it should be intuitive information. For example, taxonomy should not be long text because it will clutter the comparison result.
3. 3D game engine filter criteria selection: Every game engines should be investigated, but numerous game engines are created. Therefore, we can not add all game engines into the matrix because the comparison matrix will be cluttered. As a result, creating game engine filter criteria is a proper method to handle all game engines equally.
4. Choosing 3D game engine: This task is to study all game engines, then we screen out many unqualified game engine by using the filter criteria. For example, if some game engines have very small set of features or no high-quality games are implemented from those engines, they should not be evaluated because they are much worse than other game frameworks. These chosen game engines are columns of comparison matrix.
5. Research each chosen 3D game engine: Many chosen game engines are investigated in this task. This inspection focuses on chosen comparison criteria. As a result, the comparison matrix is completed.

Furthermore, before this comparison matrix is used, we will evaluate this comparison matrix by creating one demo application. Company will give the demo requirements. Then, comparison matrix will be used to decide optimal 3D game engine. After that, we will implement demo application using the chosen 3D game engine. While we use comparison matrix ourselves, we will know benefits and drawbacks of this comparison matrix.

# 2 Previous Work

This chapter provides background knowledge for reader. Additionally, the previous game engine comparison works are explained.

## 3D Game Engine Comparison

Today, there are 334 game engines listed in DevMaster.net database (DevMaster.net, 2012). This website contains much information about game engines, but this information is not separated between mobile and other platforms. Therefore, information in this website is not specific for handheld platform. Many functionalities of 3D game engine are different between mobile and other platform because the performance of Graphics Processing Unit (GPU) in mobile and other platforms are extremely different. In addition, OpenGL ES and mobile DirectX in mobile have lower number of functionalities than OpenGL and DirectX. For example, real-time Ambient Occlusion, 3D graphics algorithm that approximates dynamic advanced lighting, but this technique in Shiva3D is not available in mobile device (Tobias Ritschel, 2009).

In the medical field, there is an evaluation research for simulated surgical training (Stefan Marks, 2007). This paper focuses on choosing 3D game engine to implement a simulated surgical training system. They evaluate only Unreal Engine, id Tech 4, and Source Engine because other game engines do not fulfill their requirements. They choose these engines by the following criteria. First, they neglect game engines that are in an early development process, not developed, or not maintain anymore. Secondly, they disregard game engine without sound and other essential features. Third, they reduce the complexity of implementation by choosing some game engines, which has an editor. Thus, they do not need to set up other tools or convert resources from other software. Additionally, they choose editing, content, and gameplay as comparison criteria, but they explain comparison result in long text. These long texts do not summarize the result but explain only what they did and what are the problems of that engine. Therefore, the comparison result of this paper is hard to use.

The usage of 3D game engine is not limited to game industry, but it can be utilized for other working field. Mohd Fairuz Shiratuddin uses 3D game engine to teach Architectural Engineering students instead of Computer-Aided Design and Drafting (CADD) tools (Mohd Fairuz Shiratuddin, 2007). They choose BuiltITC4, which is based on C4 Engine. They did not compare many game engines, but they ask some questions to their students after students use this engine. These following questions indicate their interests about game engine.

1. Performing tasks is straightforward: World Editor can greatly help 3D virtual environment designer.
2. Organization of information: Menu layouts, icons, and terms of the engine should not be unfamiliar for students.
3. Software speed: Since 3D game engine creates real-time application, the frame rate should be responsive.
4. Software reliability: Software framework should not be crashed while working.
5. Learning to operate software: This topic is about how easy to learn the software. The authors believe that 3D game development tools are more powerful than CADD tools, because game development tools have many more features such as dynamic lighting, built-in special effects, and visual script editor.

Seung Seok Noh research about using 3D game engine for 3D entertainment contents (Seung Seok Noh, 2006). They make comparison table for Quake engine, Unreal engine, and Jupiter engine but they have only features, weak point, and masterpiece as comparison criteria.

Game based training is growing rapidly (Dan Fu, 2008). Today, militaries are trained by simulators (MACEDONIA, 2002). Dan Fu separates evaluation criteria for game engine technology for training into these following criteria.

1. Native scenario authoring capabilities: Scenario setting is troublesome, if no scenario editor is provided. Not only, pre-create scenarios are used in military training, but also unique scenarios for actual frontline battle.
2. Native support for synchronized communications: Even though communication is not necessary component of training simulation, but this can add team operation to the training simulator.
3. Cross platform inter-operability: This topic separates in to two parts. First, source code of the game engine should be accessed either directly, SDK, or API. Secondly, the game framework should support at least one of Semi-Automated Forces (SAF) standard such as OneSAF.
4. Capture: This criterion is about an ability to share the event such as demonstrations, scenarios, or simulation results. These results can be sent either via video or event logging. Therefore, the receiver can replay these results anywhere and anytime.
5. Modeled subject matter: Since the game based training focuses on military simulation, a 3D military model such as soldier and tank is needed.

Additionally, they choose some game engines without providing reasons. Chosen game engines are Torque Game Builder 2D, Unreal Engine, Game Bryo Element, Machinima 2.0, VBS1, OLIVE, and BigWorld.

Game engine selection methodology is created by (Panagiotis Petridis, 2010). They start by defining main comparison criteria, which are audiovisual fidelity, functional fidelity, composability, accessibility, networking, and heterogeneity. Then, they

break these categories down into many criteria. These following criteria are methodology for comparing engines in serious games.

1. Audio Visual Fidelity
    a. Rendering: These graphics can make same perceptual quality graphics as real life scene.
        i. Texturing
        ii. Lighting
        iii. Shadows
        iv. Special Effects
    b. Animation
    c. Sound
2. Functional Fidelity
    a. Scripting
        i. Script
        ii. Object Model
    b. Supported AI Techniques
        i. Collision Detection
        ii. Path Finding
        iii. Decision Making
    c. Physics
        i. Basic Physics
        ii. Rigid Body
        iii. Vehicle Dynamics
3. Composability
    a. Import/ Export Content
        i. CAD Platforms Supported
        ii. Import/ Export Limitations
        iii. Content Availability
    b. Developer Toolkits
        i. SDK/GDK
4. Accessibility
    a. Learning Curve
    b. Documentation and Support
        i. Docs Quality
        ii. Technical Support
        iii. Community Support
    c. Licensing
    d. Cost
5. Networking
    a. Client Server
    b. Peer-to-Peer
6. Heterogeneity
    a. Multiplatform Support

They use various taxonomies in comparison criteria such as “YES/NO”, “Small/Medium/Large”, and full text with description. Chosen game engines are

Quest3D, Blender Engine, Unreal Engine, and Unity Engine. Furthermore, they create summary chart for every main criteria. In conclusion of the paper, Unreal Engine surpasses other game engines.

## 3 Research Problem

Not many academic game engine comparisons have been done. Most of them compare game engine with specific scenarios such as training, or architectural design (Dan Fu, 2008) (Mohd Fairuz Shiratuddin, 2007). These objectives are different from the game application objective, so some of their criteria are altered from general-purpose game criteria. For example, military 3D model supported criterion is not important for ordinary game. As a result, some sections of their comparison table are unusable.

The flaw of these researches is that they mention only some game engines in their paper, but there are many more game engines than those. Effective comparison should take all available game engines into account. For example, unmentioned game engines might perform better than the chosen engine, but the reader does not know about them.

There is no academic game framework comparison for mobile platform. Performances of the game engine on various platforms are also different because they can not implement the same algorithm for every platform. Especially 3D graphics performance of the mobile platform can not be compared to the Personal Computer (PC) platform. In addition, the cross platform low-level 3D library, which is OpenGL or DirectX, is separated from PC API and mobile API.

Comparison matrix for 3D game engine for mobile devices will be useful contributions because no academic comparison work is done for this. Not only whole comparison matrix is useful, but comparison criteria themselves are the main contributions to comparison work also. People can adapt these comparison criteria to another new 3D game engine. Furthermore, method for choosing game engine is helpful for other people. We will increase comparison knowledge of the 3D game engine on mobile platform.

## 4 Comparison Matrix

In order to create the comparison matrix, we need to define comparison criteria, choose 3D game engine, and research for capabilities in each 3D game engine. Comparison matrix's readers can see the same functionality for every chosen

engine. Therefore, readers can compare game frameworks easily by checking criteria that they want to know.

## 4.1 Define comparison criteria

Game engines help game creators to develop a game application faster because game creator can use implemented technical functionalities and resources in the game engine (Dan Fu, 2008). Usable features and resources in game engine are valuable for game developer. Game engine not only helps game implementer create game faster, but it also improve quality of the game. For example, game engine's implementer uses current optimum 3D graphics algorithm for making Ambient Occlusion. Game creators can use this Ambient Occlusion algorithm on their game while they do not understand how it works. High quality game engines have numerous features. The most important functionalities are the most use components. Game engine technical functionalities are categorized by programming libraries. For instance, 3D graphics functionalities are embed in 3D graphics library. In addition, resources in game engine consist of graphics assets and example projects. Available resources are extremely useful for game creator. For instance, UDK game engine comes with First Person Shooting (FPS) game project; therefore game developer can start making FPS game extremely fast. In conclusion, programming libraries and available resources are comparison criteria.

Additionally, game developers are the main target user of this thesis. Therefore, usability of game engine is another important comparison criterion. Usability includes ease of learning, efficiency of use, memorability, error frequency, and subjective satisfaction (Jakob Nielsen, Prioritizing Web Usability, 2006). 3D game engine should be intuitive and easy to learn because new game developer can start implementing game with engine faster. Available resources help game developer learn the engine faster; so game engine resources criterion is in "ease of learning" criterion. Tools of 3D game engine can improve efficiency of working process and also reduce error frequency. Standout 3D game engines have many powerful tools. These tools can improve workflow and reduce working time. Hence, these tools can reduce the cost of 3D game. 3D game engine is not the only required component for developing a 3D game, but implementer needs 3D modeling tools also. Not all of the 3D game engines support all 3D modeling tools. There are many 3D modeling tools. Some 3D modeling tools are extremely expensive, but some tools are free. Some companies are expert on specific 3D modeling tools, so they prefer to use game engine that support for that 3D modeling tools. Since, there are many aspects of 3D modeling tools, we separate it from other comparison criteria topics. In conclusion, learning curve, tools, and supported 3d modeling tools are comparison criteria.

For investor point of view, they choose the most worth game engine for them. Cost, income, and quality of game engine are necessary information for them. Cost of using game engine is cost and licensing of the game engine itself. This might be the

largest investment of the game. Income of game comes from selling the game or services. Target customer of game and services can be categorized by deployment platform. For example, if target customer group is iOS's users, game should be run on iOS platform. Furthermore, development platform is also important because some company has only PC or Mac. So they do not need to pay for extra development environment. Therefore, price, development platform and deployment platform are comparison criteria.

All main criteria are programming libraries, tools, learning curve, supported 3D modeling tools, deployment platform, and price. These criteria will be broken down to more specific criteria. The most useful comparison criteria are the most use functionalities of game engines. These criteria's result of each game engine should be different; otherwise they are useless comparison criteria. For example, the capabilities of modify vertex shader and pixel shader are already exist in two main 3D graphics libraries which are OpenGL and DirectX. Therefore, these should not be comparison criterion.

Some previous comparison works define very specific criterion topics for their scenario such as interoperability, modeled subject matter and editing (Stefan Marks, 2007) (Dan Fu, 2008). These criteria are useless for other scenario. For example, a reader of their paper knows that one game engine has very good interoperability, but they do not know about specific aspect of their interoperability by reading the comparison result. While reader does not know about detail of the comparison result, they can not adapt it to their scenario. On the other hand, if they use "supported Semi-Automated Forces (SAF) format" as comparison criteria, reader will immediately understand that this criterion is for SAF format only. Additionally, the taxonomy of the comparison criteria should be more informative than yes or no if possible. The taxonomy of SAF criterion can be a list of supported SAF format name for each engine. If the reader wants to use some SAF format, so they will know which engine is plausible for them.

### 4.1.1 Programming Libraries

3D game engine is implemented by assembling many programming libraries together. Common programming library collection for 3D game engine consists of 3D graphics, physics, collision detection, I/O, sound, AI, and network library (Panagiotis Petridis, 2010). Collision detection is included in physics engines, because these physics engine simulates real-world physics. Other than these libraries, modern and powerful game engine embeds more libraries such as special effect, and video library. These libraries have different capabilities, which help game developer in various ways. Finally, 3D graphics, physics, sound, AI, network, special effect, and video libraries are included in programming libraries criterion.

#### *4.1.1.1 3D Graphics*

The most visible feature of game is graphics. Therefore graphics is one of the most important module for game. Unlike 2D graphics, 3D graphics has countless rendering algorithm from extremely fast to extremely slow. Since, game is interactive application,
real-time graphics algorithms are suitable for game application. The fastest graphics techniques that can produce the most beautiful graphics are used in many games. These techniques are valuable for 3D game engine.

Most mobile devices use OpenGL ES programming library to compute 3D graphics, thus 3D graphics on mobile phone are limited by OpenGL ES. This means that most of 3D game engine can do same thing, but major difference is that some engines have some 3D graphics technique built in with engine. So, users do not have to implement some techniques by using OpenGL ES that is very complicated.

Most used 3D graphics techniques for 3D game should be available in 3D game engine. 3D graphics comparison criteria should not be very basic features; otherwise these criteria can not separate between high quality and low quality game engine. Most used 3D graphics techniques are implemented in many high quality games. High quality games prove that some 3D graphics techniques are appropriate for mobile platform. So, We can find these efficient features by investigating high quality 3D game on mobile device.

One of the best graphics game on mobile platform is Infinity Blade. Infinity Blade wins WWDC Apple design awards (WWDC Apple Design Awards, 2011). This game shows capabilities of 3D game engine and new mobile devices. Infinity Blade's implementers optimize everything that they can. They use per pixel specular lighting for characters with one directional light (Epic Games, 2011), but it is only one draw call for this because they use optimized bumped specular shader. Moreover, there is no real time lighting on environment. They use baked lighting that includes global illumination, ambient occlusion, baked-in normal maps, and custom painted details. This baked lighting is very fast to compute because it already preprocesses textures with lightmap. Consequently, high quality 3D graphics can run on new mobile devices with these graphics techniques.

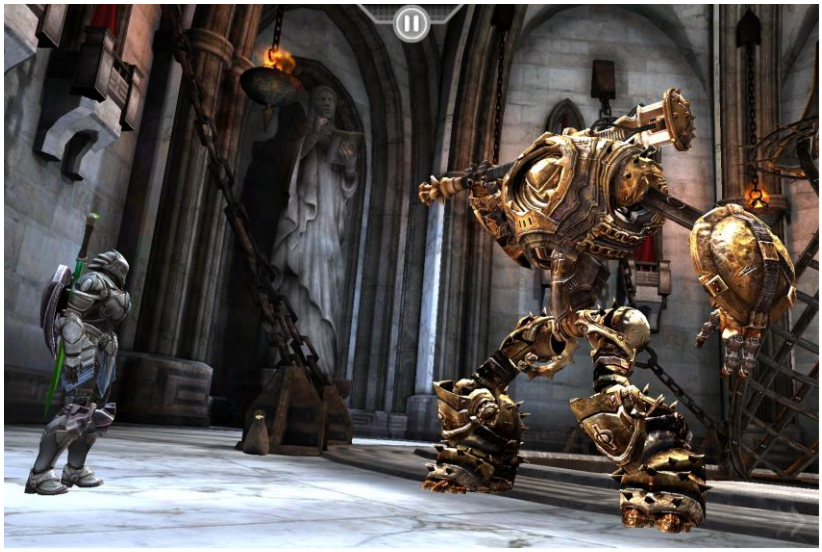

**Figure 4-1 Infinity Blade gameplay screenshot. Real lighting on character and fake lighting on environment.**


**Figure 4-2 Lightmap with and without texture.**


**Figure 4-3 Baked normal with and without texture.**

**Figure 4-4 Custon modulate with and without texture.**

Furthermore, this game is implemented on top of Unreal Engine 3. This 3D graphics result is quite similar to UDK game engine's demo application, which is Epic Citadel. For this demo, they use parallax mapping and normal mapping on different models (Tschida, 2010). For lighting, they use Global Illumination preprocessing technique to make realistic light in the scene. Moreover, implementers create reflective marble surfaces and cobblestone roads effect by using dynamic specular lighting with texture masks technique. Environment mapping technique is used to create reflection floor effect. This demo does not have any characters but they have dynamic movement trees, which is possible by implementing vertex deformation and skeletal animation. Additionally, lens flares and light coronas are added to this demo.


**Figure 4-5 Parallax mapping and normal mapping with lens flares and light coronas in Epic Citadel.**

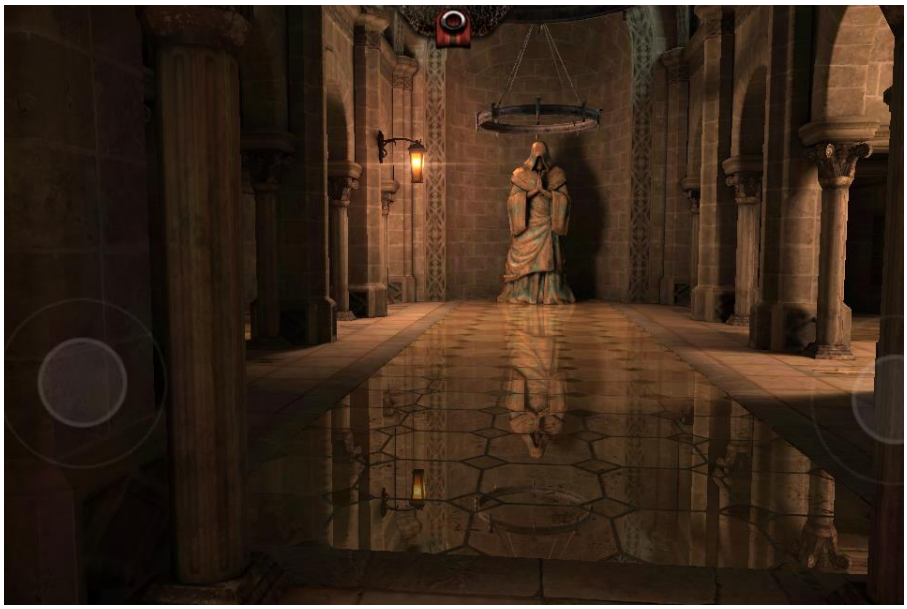

**Figure 4-6 Reflective floor and reflective marble surface on statue.**

After investigating high-level 3D graphics game on mobile devices, 3D graphics techniques that are appropriate and fast enough for mobile devices include Bump Mapping, Normal Mapping, Parallax Mapping, Static Global Illumination, and Ambient Occlusion. As we stated before, all of these techniques can be implemented on OpenGL ES, but some techniques need lots of work to be done. So, it is worthy to have these techniques ready in 3D game engine. In conclusion, each of these 3D graphics technique is an evaluation criterion. The following are evaluation criteria specification.

| | |
|---|---|
| Bump Mapping | Value: “o” = yes, “x” = no |
| Normal Mapping | Value: “o” = yes, “x” = no |
| Parallax Mapping | Value: “o” = yes, “x” = no |
| Global Illumination | Value: “o” = yes, “x” = no |
| Ambient Occlusion | Value: “o” = yes, “x” = no |

#### *4.1.1.2 3D Physics*

3D Physics create dynamic object in 3D world. It simulates real world physics into 3D virtual world. Additionally, it needs to calculate force and collision detection of each 3D model. There are so many types of object material in real world like rubber, metal, gas, fluid, cloth, and glass. These materials’ effect can be achieved by different 3D graphics algorithm. Therefore, physics library can not simulate all of material, so physics library needs to choose which material they will support. For example, rubber bounce effect can be simulated by Soft Body technique (Miao Song, A framework for dynamic deformation of uniform elastic two-layer 2D and 3D objects in OpenGL, 2008). In addition, physics engine is very important to simulate real world effects for 3D game engine. For example, when bomb explodes, object around

should be diffracted. Physics feature is extremely complicated. Therefore so many 3D game engines use other Physics Engine to achieve these dynamic simulations. There are many different algorithms for different physics library.

There are many 3D physics engines, but PhysX, Havok, Open Dynamics engine (ODE), Newton Game Dynamics, and Bullet are popular physics engine (Wikipedia, 2011). So, information for this evaluation criterion should be name of the physics engine that they use. Thus, name of physics engine can show many different capabilities about dynamics object in each 3D game engine because each physics engine has different set of features. But overall result PhysX is the best physic engine compared to other engine (Adrian Boeing, 2007). Thus, 3D game engines that use PhysX will be marked as green color because it has an advantage for them to use PhysX physics engine. Some 3D game engines that implement their own physics engine will be marked as "o". Then, 3D game engine that does not use physics library will be marked as "x".

**Figure 4-7 This scene includes thousands of dynamic pipes using PhysX physics engine.**

#### *4.1.1.3 Artificial Intelligence*

Artificial Intelligence (AI) is important for most of the games because monsters or computer's players need to use AI in order to fight with human automatically, but there is no common AI that can be used with most of the game because there are countless types of game. Additionally, some 3D game engines implement AI about

path finding and character behavior. This AI technique is mostly used for some game genre like Role-Playing Game (RPG), First Person Shooter (FPS), and Real Time Strategy (RTS). Countess AI techniques can be used for game depending on gameplay itself. There is no main category of AI techniques that cover all AI algorithms for game. Consequently, information of this evaluation criterion is “o” for yes or “x” for no.

#### *4.1.1.4 Network*

Network in this case is not the network to browse Internet but it is network to communicate between each game client. Network library is required in multiple device multi-player game because each multi-player games client needs to communicate with other client. There are many network models of multiplayer game, but the most complicated network model for multiplayer game is massive multiplayer online game (MMO) because numerous people can play on the same virtual world at the same time. For example, World of Warcraft is one of the most successful MMO game. It can support more than 1 million concurrent users (Schramm, 2008). Consequently, information of this evaluation criterion is “o” for yes, “x” for no, and “MMO” for network library that support MMO game. Additionally, MMO supported library will be marked with green color.


**Figure 4-8 This image shows one event in World of Warcraft games. There are thousands of characters, which are real human playing this game in same place.**

#### *4.1.1.5 Particle Effects*

Particle effects are special technique to reproduce effect from real world. By using particle effect, designer can copy effect like fire, trails, explosion, dust, fog, and cloud conveniently. Numerous particles can be created from set of emitters, so particle effects mimic behavior of some phenomenon like dust, fog, and cloud. Particle effects can also copy more advanced effects like explosion with many configurations. Consequently, information of this evaluation criterion is “o” for yes or “x” for no.

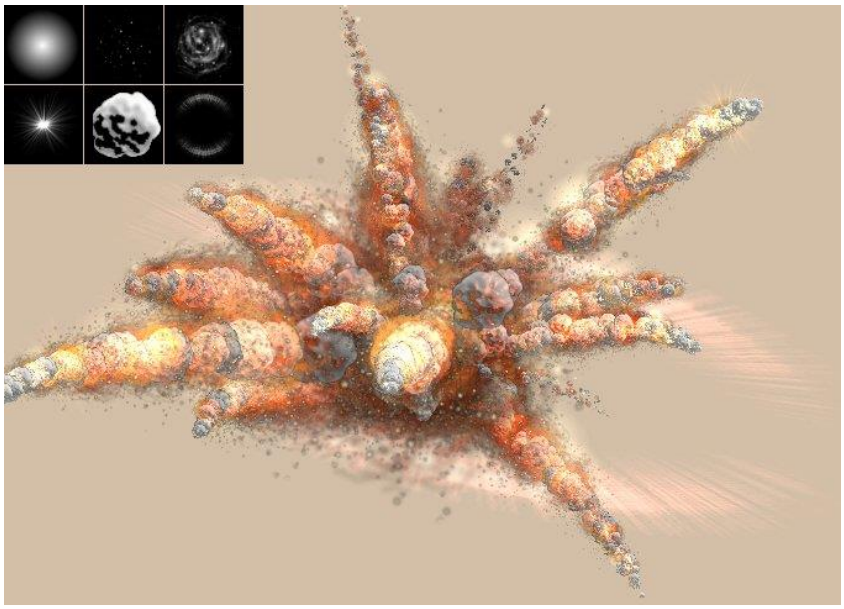

**Figure 4-9 This is explosion effect created from many particle effects. Six images on top left are used for each billboard particle in this explosion**

#### *4.1.1.6 3D Audio*

3D audio is different from normal audio because it has 3D position in 3D world. So, it is like real world that every sound has its own position. If players are near to the audio source, the sound will be louder. Only 3D sound position feature is not good enough for advanced game engine. So, some 3D game engines use powerful audio library in order to edit sound and optimize performance. Moreover, some 3D game engines support 2.0, 4.0, and 5.1 audio channels, therefore player can hear surrounded sound. Performance of real-time application like game is very important, so compression and optimization of audio source is necessary. Furthermore, streaming audio from Internet is another possible feature for audio library also. As a result, information of this evaluation criterion is name of the audio engine, “o” for yes, or “x” for no.

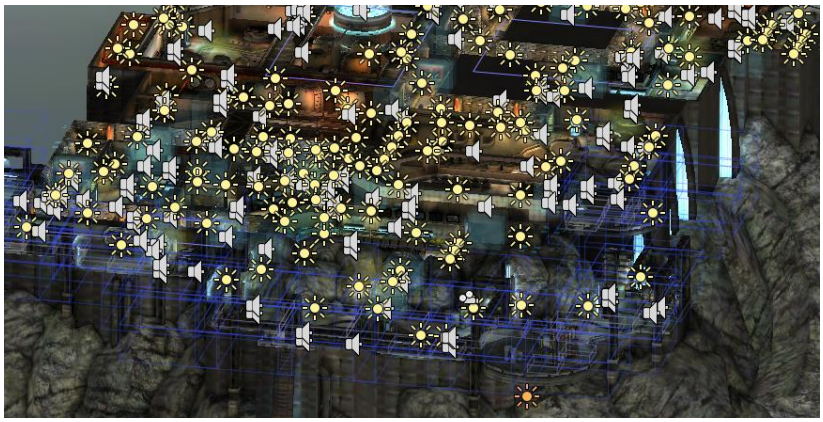

**Figure 4-10 This is scene in Unity3D editor, which is use fmod library. Each speaker icon is position of one sound in virtual world.**

#### *4.1.1.7 3D Video*

3D video is not like normal video at all because 3D video can be played on 3D texture. So, texture can move and rotate with the 3D model. Additionally, game implementer can use shader with video texture [Figure 4-11]. There is no separated 3D video engine but there are differences between video compression algorithms. Video compression algorithm is very critical issue because video usually is very large file. All game engines need video compression algorithm that can compress video to small size but still preserve high video quality. So, information of this evaluation criterion is name of video codec, "o" for yes, or "x" for no.

**Figure 4-11 This is one example of using video texture and pixel shader in one scene.**

### 4.1.2 Usability

Usability includes ease of learning, efficiency of use, memorability, error frequency, and subjective satisfaction (Jakob Nielsen, Prioritizing Web Usability, 2006). 3D game engine should be easy to learn because new developer can start working with

the engine faster. Memorability is similar to learning, because users need to know the system and remember engine's user interface before they can use it. Tools of 3D game engine can improve efficiency of use and also reduce error frequency. High-end 3D game engine has many tools. These tools can improve workflow and reduce working time. So, these tools can reduce the cost of 3D game. Not only 3D game engine is required to develop new game, but implementer needs 3D modeling tools also. Not all of the 3D game engines support all 3D modeling tools. In addition, these 3D modeling tools are expensive so it is one of the overall investments of game. In conclusion, ease of learning, and efficiency of use are main evaluation criteria for usability.

#### *4.1.2.1 Ease of learning*

Ease of learning does not have direct indicator to measure. One way to measure this criterion is to investigate how developer learn game engine. There are many resources that created from the owner for user such as documentation, tutorials, and examples. These resources are necessary for game developer; otherwise game developer can not understand the engine. Not only 3D game engine's creator can create tutorials and examples, but game engine's user can make them too. Therefore, community of each 3D game engine is very important also. For examples, Unity3D engine has many websites teaching Unity3D. So, there are much more learning resources. These communities means that many people are using the engine, so 3D game engine's user can ask a question to them. Moreover, Unity3D also has asset store which is market place for Unity project. These stores extremely help developers, because they can buy the codes and assets, which already built on Unity3D game engine. Each of these website has their own community. All good 3D game engines always have their tutorials, examples, and communities. We decide to show "1" to "5" indicate the quality and amount of tutorials, examples, and communities.

Additionally, each programming language has different learning curve, but sometimes programmers already understand that programming language, that engine will be easy for them to implement. So, taxonomy of programming language criteria should be name of programming language, because this criterion depends on comparison matrix's user. Furthermore, if user has a problem working with engine, the best way to solve the problem is to ask the engine's support. Support from 3D game engine is beneficial for user, so it should be one of our evaluation criteria. Most of the engines do not have these features without pay more money. Therefore, we indicate as "o" for has support and "x" for no support.

#### *4.1.2.2 Efficiency of Use*

Efficiency of use is the next step from learning. Well-designed game engine should provide efficient workflow; because game developers have to work with workflow designed by game engine's creator. Game engine's workflow is defined by set of tools. This is not whole workflow, but it defines a part of whole workflow. Not only efficient workflow can reduce the working time, but well-designed tools can also reduce working time. Most of the time, game development workflow needs to use 3D modeling tools in order to create 3D resources. In summary, tools and supported 3D modeling tools are main criteria of this topic.

#### 4.1.2.2.1 Tools

3D game engine is not only set of library but it also has set of development tools. Basically, all resource files for 3D game engine should be editable by development tools because designers can see the real result in the 3D game engine. So, designer does not need to export and import 3D assets to 3D game engine so many times. Consequently, development tools can optimize workflow of the 3D game development. Moreover, programmers need some tools for development such as programming editor, debugger, and profiler. There are many types of 3D resources such as map, animation, material, AI, GUI, particle effect, cut scene, and audio. So, each one of these 3D resources is one editor. But these editors might combine into few tools in 3D game engine because it is easier to work with few applications.

##### *4.1.2.2.1.1 Map Editor*

Map editor is used the most because it combines 3D resource together in order to create 3D world. Game player will play within 3D world all the time. So 3D world should be very beautiful. Designer needs to make beautiful scene, but it is very hard to create game compatible 3D world from 3D modeling tools, because 3D modeling tools are designed for movies industry. 3D world for movie industry is totally different from 3D world in game. Game is real-time application with more than 30 frames per second, but 3D movies can pre render every frame with 11.5 hours per frame in "Cars 2" (Terdiman, 2011). All 3D game engine do not work closely with 3D modeling tools, so some settings of 3D world in 3D game engine might be different from 3D modeling tools. This causes the different in 3D world after import scene. Furthermore, if 3D game engine does not have map editor, it will need to export and import so many times because designer cannot see the actual result from 3D modeling tools. The problem is that the 3D world might be different result on many applications. If 3D game engine has map editor, designer can change some small thing so fast. Additionally, there is some technique that should not import 3D model directly like terrain. Terrain can be created by 2D gray image, so there is no need to save so large 3D model for terrain. If map editor tool can create terrain by 2D gray image, it will reduce the size of 3D game. Consequently, information of this

evaluation criterion is “o” for game engine that has map editor or “x” for game engine without map editor.

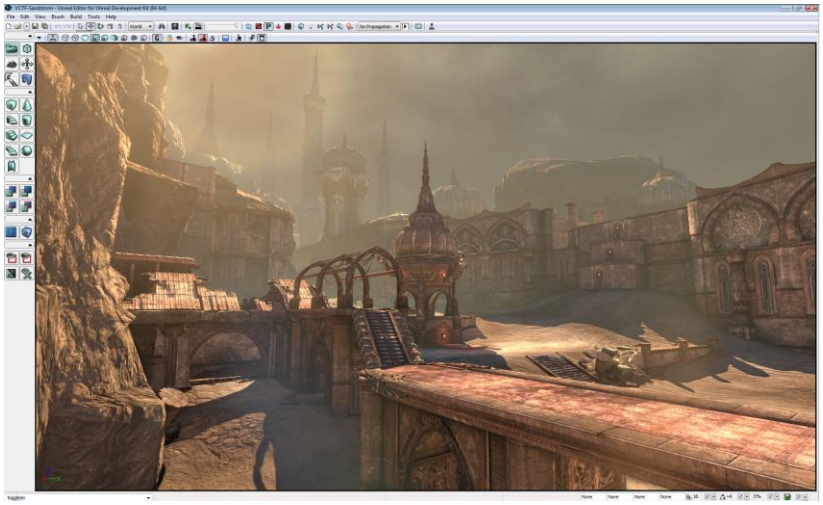

**Figure 4-12 This is map editor in UDK engine.**

#### *4.1.2.2.1.2 Animation Editor*

Animation asset can be created by 3D modeling tools, but it might not be perfect because settings of 3D modeling tools and 3D game engine might not be the same. Moreover, there are some extra features of each 3D game engine that 3D modeling tools cannot do. For example, character animation can be linked with 3D physics, so character can interact with 3d object by using 3D animation. Furthermore, 3D modeling tools do not have some optimizations that 3D game engine can do. In conclusion, animation editor inside 3D game engine is the best solutions for create animation for 3D game engine. Consequently, information of this evaluation criterion is “o” for game engine that has animation editor or “x” for game engine without animation editor.

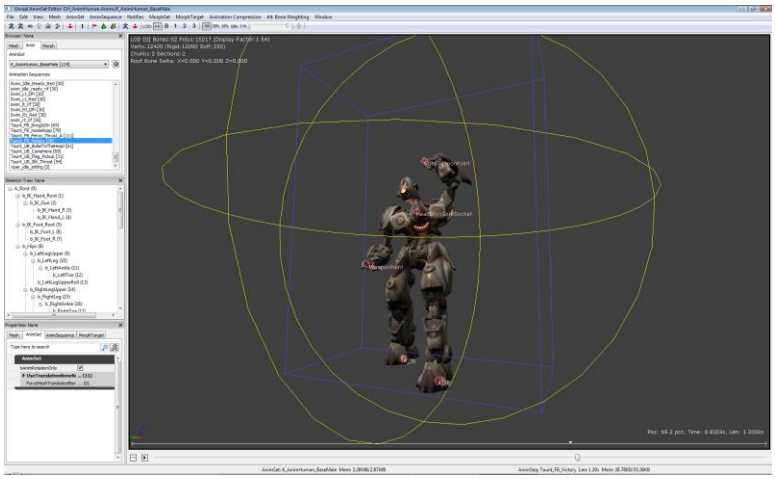

**Figure 4-13 This is animation editor in UDK engine.**

#### *4.1.2.2.1.3 Material Editor*

There are countless shaders because shader is programmable. But shader codes are different between each 3D game engine and also different between 3D game engine and 3D modeling tool. So designer cannot import shader from 3D modeling tool to 3D game engine directly. Additionally, material editor can show the result of the shader. Therefore, designer can create specific effect so easily. Some 3D game engine has many ready-made materials; so game designer can use it without writing any code. Furthermore, some game engine like UDK has complete real time visual interface for editing material, thus it is much easier for designer. Consequently, information of this evaluation criterion is “o” for game engine that has material editor or “x” for game engine without material editor.

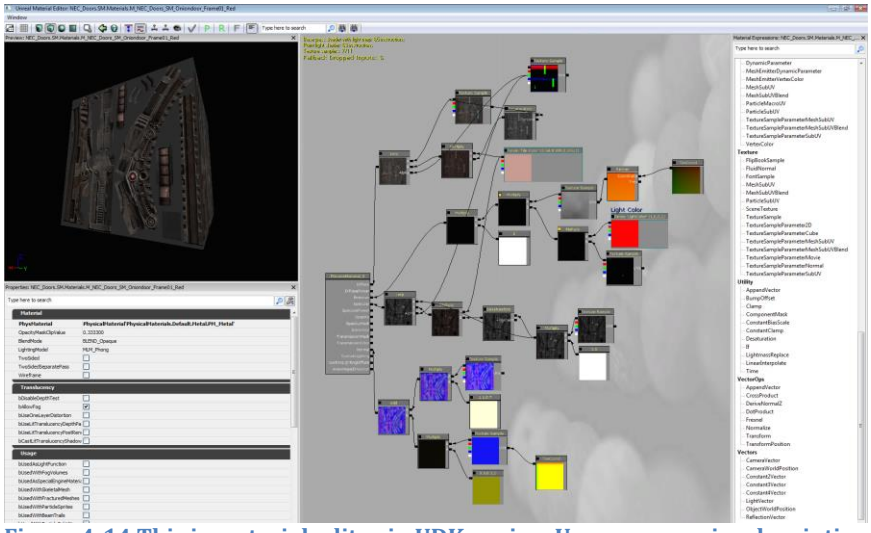

**Figure 4-14 This is material editor in UDK engine. User can use visual scripting editor like this instead of writing shader code.**

#### *4.1.2.2.1.4 AI Editor*

Most used AI technique is a navigation system, so this system is built-in technique for some 3D game engines. AI editor can help designer to put waypoint or spawn point of AI character. After game designers set waypoint and spawn point in the 3D game engine, they can run the game to test AI very easy. Consequently, information of this evaluation criterion is “o” for game engine that has AI editor or “x” for game engine without AI editor.

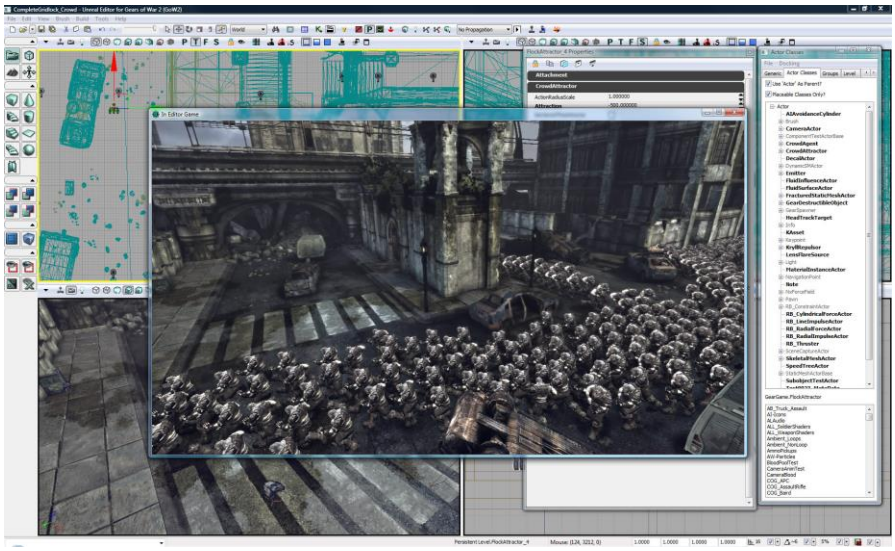

**Figure 4-15 This scene shows flocking of enemy characters from UDK engine.**

#### *4.1.2.2.1.5 GUI Editor*

Graphics user interface (GUI) is a 2D user interface. Even though the game is 3D, but countless 3D game has 2D user interface. It uses for main control of the application such as menu screen and options screen. GUI editor helps designers to make GUI because they can put each control wherever they want on the screen. It includes many ready-made GUI control like slide bar, checkbox, and button, so game developer can use them very conveniently. Benefit of these ready-made GUI is that users are familiar with these controls. Consequently, information of this evaluation criterion is “o” for game engine that has GUI editor or “x” for game engine without GUI editor.

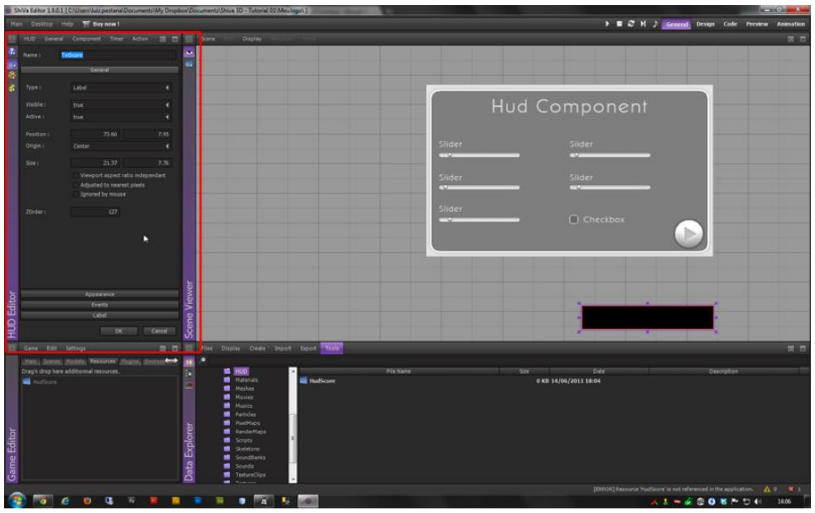

**Figure 4-16 This is Shiva 3D editor, which has GUI editor features.**

### *4.1.2.2.1.6 Particle Effect Editor*

Particle effect is quite complicated because it combines material, emitter, and particle together. Additionally, some advanced particle effects need to combine many different particle effect in order to create high-level effect. There are so many parameters for each particle effect, which are time based. It use fast graphics algorithm with thousands of particles. It can imitate the slow effect with faster computation time. So, real-time result of particle effect is very necessary for creating complicated effect. Consequently, information of this evaluation criterion is “o” for game engine that has particle effect editor or “x” for game engine without particle effect editor.

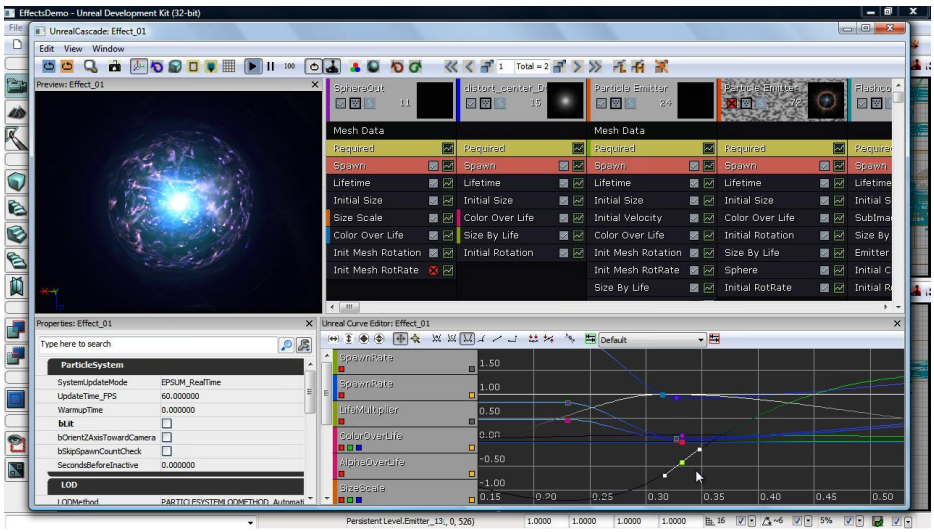

**Figure 4-17 Particle effect editor from UDK engine. Texts on the right show parameter of current effect. Graph on bottom right show value of each parameter within time range. Left top image is the real-time render of current particle effect.**

#### *4.1.2.2.1.7 Cut Scene Editor*

Cut scene editor is cinematic tool for 3D game engine. Designer can create cut scene very easily by using their game resources because cut scene will render the 3D world in real-time. Usual map editor can organize cut scene as well, but it needs a lot more work to create cut scene from map editor without cut scene feature. Cut scene editor controls cameras, 3D models, 3D animations, particle effects, and pixel shaders based on time and key frame. Therefore, real-time movie scene can create interactive movie scene. Moreover, editor can also pre render the scene to video file. So, movie scene can be created beautiful and complicated scene from PC, which is more powerful than mobile devices. Then, game plays the pre-render movie on mobile devices instead of renders the scene in real-time. Information of this evaluation criterion is “o” for game engine that has cut scene editor or “x” for game engine without cut scene editor.

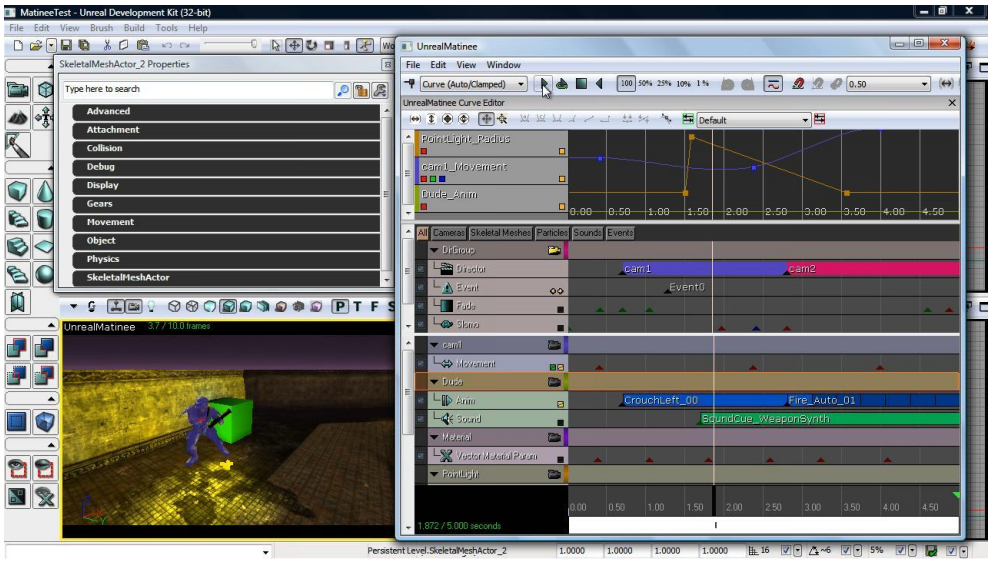

Figure 4-18 This is cut scene editor of UDK engine.

#### *4.1.2.2.1.8 Audio Editor*

Audio editor is not only 3D positioning audio on the scene, audio editor also helps designer to adjust ranges of sound in 3D world. This editor helps a designer to edit attenuation curves. In addition, the editor can modify or add sound effect to original audio file. For example, hard surface reflects sound, so designer can add echo effect to the sound without changing the original audio source. Information of this evaluation criterion is “o” for game engine that has audio editor or “x” for game engine without audio editor.

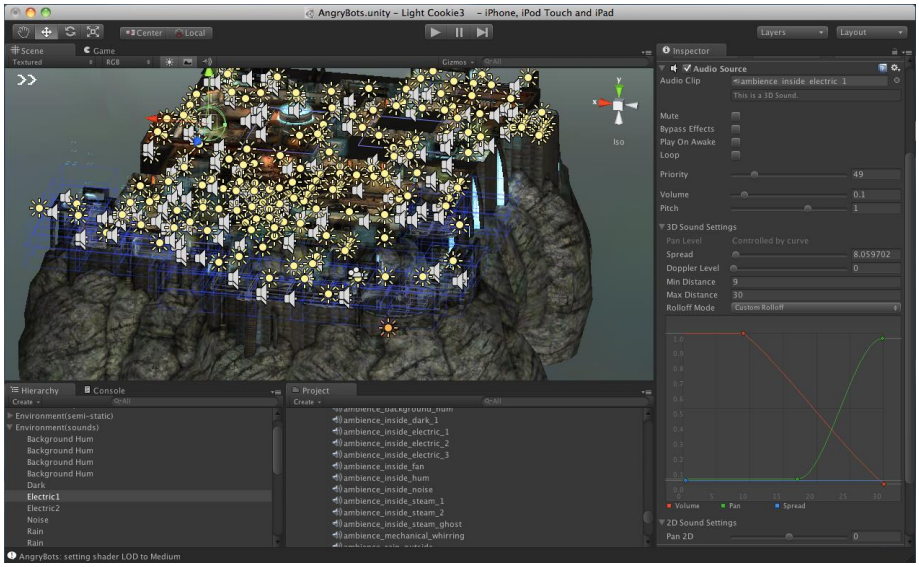


**Figure 4-19 This is audio editor of Unity3D engine**

### *4.1.2.2.1.9 Programming Editor*

Programming editor is mostly used by programmer because the editor is the main editor that programmer can add a gameplay logic and an event. Nowadays, there are many powerful editors like Eclipse, NetBeans, Visual Studio, XCode, and Mono Develop. Some 3D game engines use these programming editor, but some engine use their own programming editor. For example, UDK engine has Unreal Kismet, which is a visual scripting editor, but it can do only basic game logic. If implementers need to create complicated logic, they have to use Unreal Script with other programming editor. There are many advantages and disadvantages for each programming editor. We can not judge the best editor because advantages and disadvantages depend on user's skill also. Information of this evaluation criterion will be name of programming editor, and "o" for in house editor.

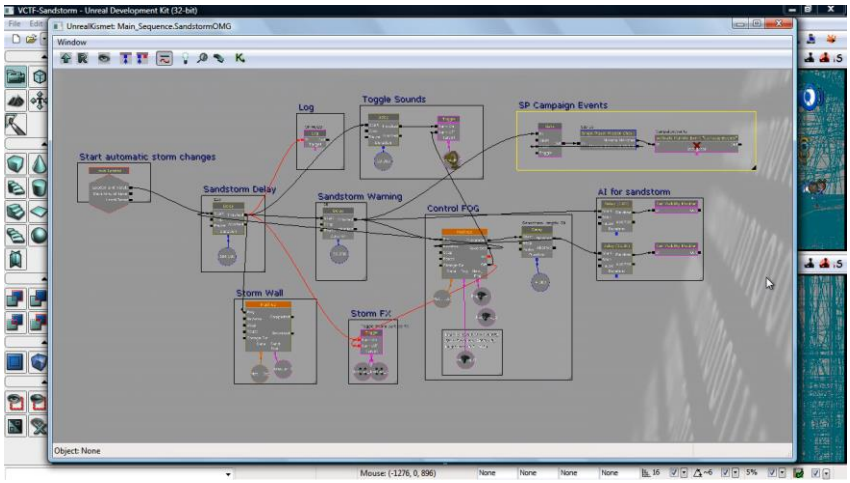


**Figure 4-20 This is Unreal Kismet, visual scripting editor of UDK engine.**

#### *4.1.2.2.1.10 Debugger*

Debugger usually is combined with programming editor, but debugger is a very useful feature of programming editor. It should be separated evaluation criteria. This editor helps programmers a lot while developing an application. The debugger can stop the application at a specific line of code and programmers can check all the current value at run time. So, programmers know the source of the bug and they can fix bug efficiently. Information of this evaluation criterion is “o” for game engine that has debugger or “x” for game engine without debugger.

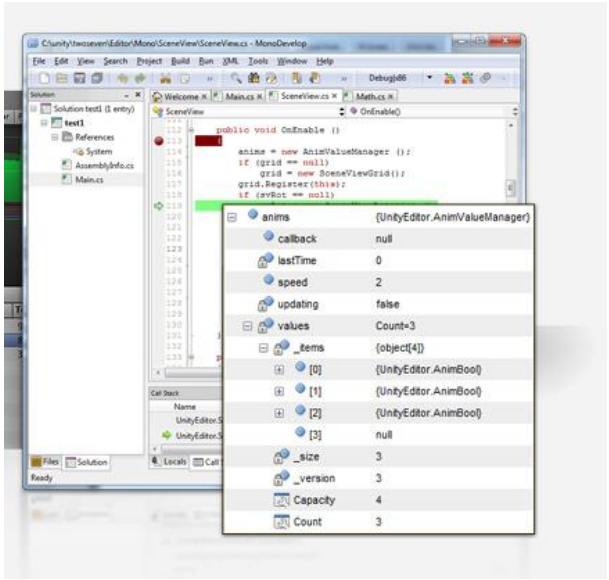


**Figure 4-21 This Mono develop is use as debugger for Unity3D engine.**

#### *4.1.2.2.1.11 Profiler*

Profiler is really needed for real-time game because everything in game affect frame rate of the application. The hardest part of 3D game is to optimize graphics and logic, such that user can play the best 3D graphics for their devices. Profiler shows bottleneck calculation of the game, so implementers know the problem, and then game developers can fix it. Otherwise, this bottleneck would be very hard to investigate through whole computation of the game. Information of this evaluation criterion is “o” for game engine that has profiler or “x” for game engine without profiler.

**Figure 4-22 This is profiler of Unity3D engine.**

#### 4.1.2.2.2 Supported 3D Modeling Tools

3D modeling tools are designed for creating 3D animation movies. So it has all the features to make movies. These tools can make 3D animation movies from scratch, so they are capable of making 3D model, 3D animation, 3D effect, particle effect, and render movies. 3D game does not use all features from the tools.

The most famous 3D modeling tools for 3D game are MAYA, 3D Studio MAX, Blender, and SoftImage (XSI). Not every 3D game engine supports all of these tools. It is too complicated to show every detail of these tools with 3D game engine, so we will separate each 3D modeling tools to one of evaluation criteria. Most of 3D designers are specialized with only one 3D modeling tool, so it is worthy for them to know about which 3D game engine they can work with. So, information of this evaluation criterion is “o” for yes or “x” for no.

### 4.1.3 Development Platform

The main computer OS are Windows, OSX, and Linux. Market share of Windows, Mac OSX, and Linux are 68.74%, 7.78%, and 6.81% (Zachte, 2012). Other than these three platforms have less than one percent market share. Some game engine

support iOS platform, but cannot run game engine on Mac OSX. These engines have disadvantage, because all application that can publish on Apple store must use Mac OSX. Windows, Mac OSX, and Linux are evaluation criteria. Information of this evaluation criterion is “o” for yes or “x” for no.

### 4.1.4 Deployment platform

This report focuses on mobile platform, so the main mobile platforms are the most important. Top four mobile operating system (OS) are SymbianOS, iOS, Android, and Blackberry OS [Figure 4-23]. These four mobile OS cover more than 80 percent of all mobile OS in the world, but not all of these platforms are suitable for 3D game. (Techcrunch, 2011) states that Nokia will make Windows phone instead of Symbian phone, so SymbianOS does not have competitive future. Additionally, first Blackberry phone is released on year 2002, which is very fast compared to iOS and Android, but number of available applications on Blackberry OS is much less than iOS and Android [Figure 4-24]. This statistics shows that not many games are developed for Blackberry OS, and not many Blackberry OS’s users play game. Additionally, Windows Phone 7 is new and powerful. Windows Phone 7 Marketplace reaches 30000 applications within 11 months [Figure 4-25]. Windows Phone 7 has more applications than Blackberry OS within one year, so this platform can be one of the best mobile platforms. It cannot compare with iOS and Android today, which has more than 200000 applications. As a result, Android, and iOS are the most suitable platform for game application. We will show the entire gaming platform because some developers might want different platforms. But if which 3D game engine has both iOS and Android, they will be highlight with green color. Otherwise they will be highlight with red color.

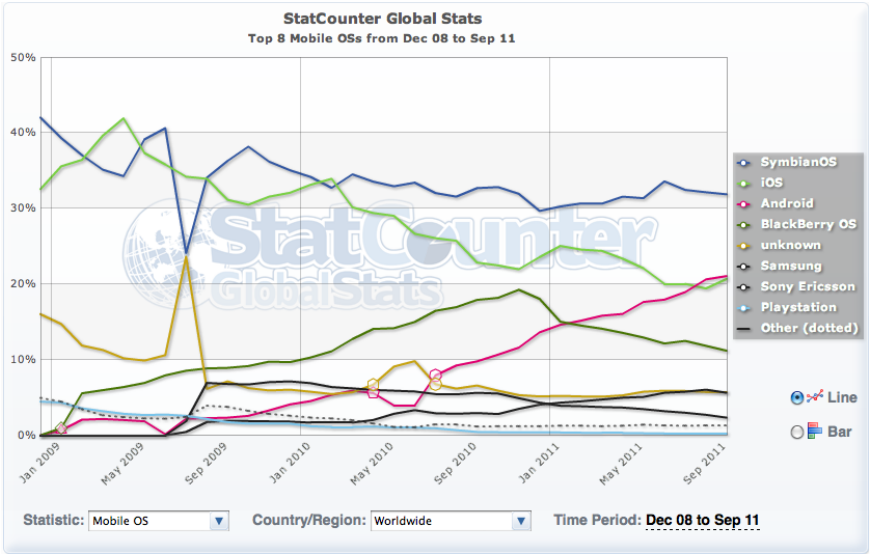


**Figure 4-23 This global statistics shows that top 4 mobile OS are SymbianOS, iOS, Android, and Blackberry OS. (StatCounter, 2011)**

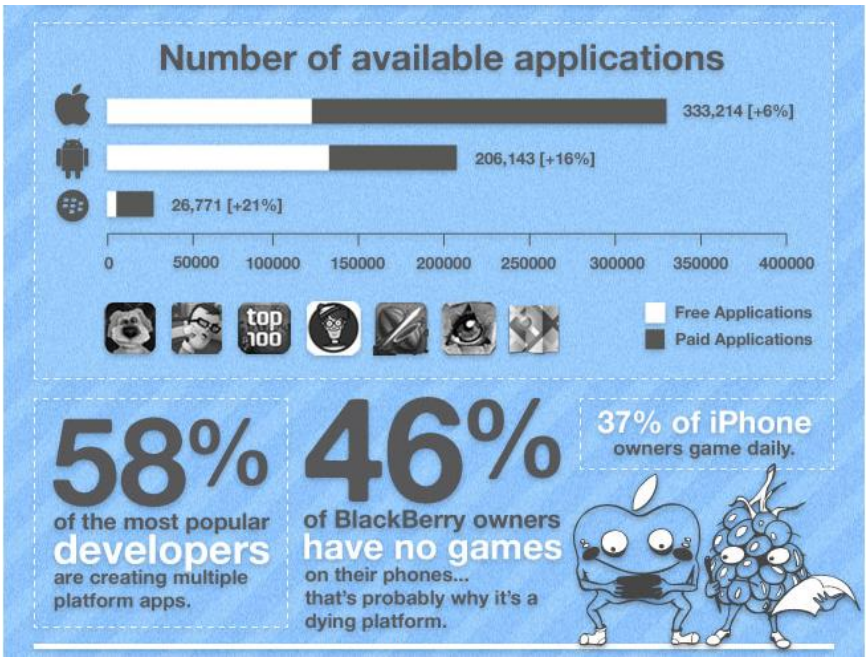


**Figure 4-24 This statistics shows number of game application available for each mobile OS. (DigitalBuzz, 2011)**

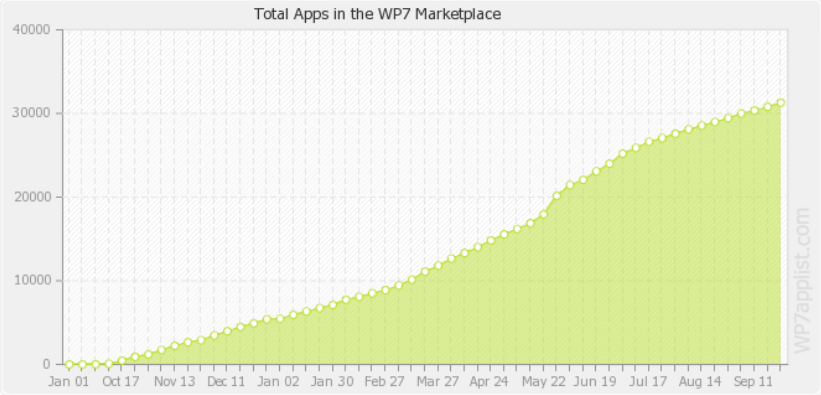


**Figure 4-25 This graph shows that number of application in Windows Phone 7 is growing rapidly. It reached 30000 applications in 11 months since October 2010. (wp7applist, 2011)**

### 4.1.5 Price

License of 3D game engine is like other software package, which has open source, free, price per year, price without royalty and price with royalty. So, price or type of license should be in comparison matrix. Information of this evaluation criterion is “opensource” for open source game engine, price for proprietary without royalty fee, price per year for year subscription, or price with royalty fee.

## 4.2 Define 3D game engine filter criteria

More and more games are successful these days. Those games use different tools and libraries in order to create game, because there are plenty of powerful programming libraries. Game developers can use these libraries to speed up their work, but the most appropriate tool for creating game is game engine, because these engines are designed for game application. This paper is interested on 3D mobile game, so we can remove all 2D game engine and non-mobile game engine.

Furthermore, 3D games use so many resources on mobile device, so only new mobile phone models can run high quality 3D games. Therefore, we will focus on only successful smart phone operating systems, which are SymbianOS, Blackberry OS, iOS, and Android [Figure 4-23]. Nokia changes their new phone to Windows phone platform (Techcrunch, 2011); therefore not many Symbian phones will be released in the future. Consequently, Symbian OS is not appropriate platform for 3D game. Additionally, number of Windows Phone 7 applications is growing rapidly [Figure 4-25]. So 3D game engine should support at least one of these four platforms, which are Blackberry OS, Windows Phone 7, iOS, and Android.

Our goal is to provide usable comparison result, because user wants to know which game engine that they will invest on. Their chosen game engine will affect the overall result of their games. We can prove that 3D game engine is valuable by checking number of released mobile game from that engine. If those games are too low quality and not many of them on mobile devices, that 3D game engine will be removed from our comparison matrix. If game engine has 3D game on mobile devices less than 3 games, it means that no one use it. As a result, it lacks of community. About open source 3D game engine, if the latest update is more than one year, it shows that it lacks of developer community. Since this game engine does not update new features. It will lose other game engine, because computer science is growing rapidly. As a result, 2D, non-mobile, mobile without successful platform, low quality, and tiny community game engines are inappropriate game engines for comparison.

## 4.3 Choose 3D game engine

Before we can choose 3D game engine, we should find existing 3D game engine list first. There are 345 game engine in Devmaster game engine database (devmaster.net, 2011), 336 game engines in Moddb (DesuraNET Pty Ltd. All Rights Reserved., 2012), and more than 150 game engines in Wikipedia (Wikipedia, 2012). After we use obvious filter like non-mobile, and non-3D, game engines passed both criteria are Unity, Raydium, Linderdaum Engine, ShiVa Engine, Esenthel Engine, Unreal Development Kit, GameStart, App Game Kit, DX Studio, SIO2, Ogre3D, Unigine, Sunburn, oolongengine, jPCT, and Marmalade. All of these game engines, Linderdaum Engine is not release full version, therefore this game engine will not be

compared. All chosen game engine other than Linderdaum Engine pass all the filtered criteria.

## 4.4 Investigate Each Chosen Game Engine

This section investigates and evaluates every comparison criteria with every chosen game engine. Most of these information are available in game engine's website. They show some features in their demo project. We consider only features that are officially built within game engine. Programming libraries, usability, development platform, deployment platform, and price are main criteria that we will explain. Since all official game engine websites indicate development platform, deployment platform, and price, we explain only programming libraries and usability.

### 4.4.1 Programming Libraries

This topic has two main categories, which are 3D graphics and other libraries. For all libraries other than 3D graphics library, we can retrieve information from game engine's website. Some information for 3D graphics is tricky to find, because they are small but important features. For example, Shiva3D support ambient occlusion but not support for mobile.

### 4.4.2 Usability

Ease of learning and efficiency of use are main aspect for this topic. About ease of learning, it includes tutorial, example, community, programming language, and technical support. There is no direct information about how many tutorials, examples, and communities. We evaluate them by check number and quality of tutorials, examples, and communities that can access without paying more money. Game engine's creator provides mostly tutorials. Communities provide few of them. Examples are available from game engine's creator and communities. Number of active forums, website, and asset store are required to determine how many communities. Programming language and technical support information are available from their website. We investigate tools topic in efficiency of use by checking all of each game engine's tools. Most of game engine's tools are embedded in one application. Each game engine's application has various editors.

## 4.5 Comparison Matrix

| Features | UDK | Unity3D iOS Pro | Shiva3D Advance | Marmalade | Dx Studio | Esenthel Engine | SIO2 | Ogre3D | Unigine | SunBurn | Raydium | oolongengine | jPCT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1. Programming Libraries** | | | | | | | | | | | | | |
| 1.1 3D Graphics | | | | | | | | | | | | | |
| 1.1.1 Bump Mapping | o | o | o | o | x | o | x | o | o | x | x | x | x |
| 1.1.2 Normal Mapping | o | o | o | o | o | x | x | o | o | o | x | x | x |
| 1.1.3 Parallax Mapping | o | o | x | x | o | x | x | x | o | x | x | x | x |
| 1.1.4 Static Global Illumination | o | o | x | x | x | x | x | x | o | x | x | x | x |
| 1.1.5 Ambient Occlusion | o | Not on mobile | Not on mobile | x | x | Not on mobile | x | x | o | o | x | x | x |
| 1.2 3D Physics | PhysX | PhysX | ODE | ODE | PhysX & Bullet | PhysX & Bullet | Bullet | x | o | o | ODE | Bullet | o |
| 1.3 Artificial Intelligence | o | x | x | x | o | o | o | x | o | x | x | x | x |
| 1.4 Network | o | o | o | o | MMO | MMO | o | x | o | x | o | o | x |
| 1.5 Particle Effects | o | o | o | x | o | o | o | o | o | x | o | o | x |
| 1.6 3D Audio | fmod | fmod | o | o | o | o | OpenAL | x | o | o | o | o | x |
| 1.7 3D Video | Bink | o | o | o | Theora | x | o | x | o | x | o | x | x |
| **2. Usability** | | | | | | | | | | | | | |
| 2.1 Ease of Learning | | | | | | | | | | | | | |
| 2.1.1 Tutorials | 4 | 5 | 3 | 4 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 2 |
| 2.1.2 Examples | 4 | 5 | 3 | 4 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 1 |
| 2.1.3 Communities | 4 | 5 | 2 | 3 | 3 | 2 | 3 | 4 | 1 | 3 | 3 | 1 | 3 |
| 2.1.4 Programming Language | Unreal Script | C#, Javascript | LUA | C++ | Javascript | C++ | C++ | C++ | C++ | C# | C | C++ | JAVA |
| 2.1.5 Technical Support | x | x | x | x | o | o | o | x | o | x | x | x | x |
| 2.2 Efficiency of Use | | | | | | | | | | | | | |
| 2.2.1 Tools | | | | | | | | | | | | | |
| 2.2.1.1 Map Editor | o | o | o | o | o | o | Blender | x | o | o | x | x | x |
| 2.2.1.2 Animation Editor | o | o | o | o | o | o | Blender | o | o | x | x | x | x |
| 2.2.1.3 Material Editor | o | o | o | o | o | o | x | x | o | o | x | x | x |
| 2.2.1.4 AI Editor | o | x | x | x | o | x | x | x | x | x | x | x | x |
| 2.2.1.5 GUI Editor | Flash | x | o | o | o | o | x | o | x | x | x | x | x |
| 2.2.1.6 Particle Effects Editor | o | o | o | x | o | x | x | o | o | x | x | x | x |
| 2.2.1.7 Cut Scene Editor | o | Animation Editor | x | x | x | x | x | x | x | x | x | x | x |
| 2.2.1.8 3D Audio Editor | o | o | o | o | o | o | o | x | o | o | x | x | x |
| 2.2.1.9 Programming Editor | Visual Studio, Kismet | Mono Develop | o | o | o | Visual Studio | iOS SDK | iOS SDK | o | o | iOS SDK | iOS SDK | o |
| 2.2.1.10 Debugger | Visual Studio | Mono Develop | x | o | o | Visual Studio | iOS SDK | iOS SDK | o | x | iOS SDK | iOS SDK | o |
| 2.2.1.11 Profiler | x | o | x | iOS SDK | x | x | iOS SDK | iOS SDK | x | x | iOS SDK | iOS SDK | x |
| 2.2.2 Supported 3D Modeling Tools | | | | | | | | | | | | | |
| 2.2.2.1 MAYA | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.2 3D Studio MAX | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.3 Blender | o | o | o | o | o | o | o | o | o | o | o | o | o |
| 2.2.2.4 XSI | o | o | o | o | o | o | o | o | o | o | o | o | o |
| **3. Development Platform** | | | | | | | | | | | | | |
| Windows | o | o | o | o | o | o | o | x | o | o | x | x | o |
| Mac OSX | x | o | o | o | x | o | o | o | o | x | o | o | o |
| Linux | x | x | x | x | x | x | x | x | o | x | x | x | o |
| **4. Deployment Platform** | | | | | | | | | | | | | |
| Mobile Phone | | | | | | | | | | | | | |
| -iOS | o | o | o | o | x | o | o | o | o | x | o | o | x |
| -Android | x | o | o | o | o | o | o | x | Some GPU supported | x | x | x | o |
| -WebOS(Palm) | x | x | o | o | x | x | o | x | x | x | x | x | x |
| -Windows Mobile 6 | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| -Windows Phone 7 | x | x | x | x | x | x | x | x | x | o | x | x | x |
| -Bada | x | x | AirPlay | o | x | x | o | x | x | x | x | x | x |
| -Simbian | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| Mobile Console | | | | | | | | | | | | | |
| -PSP | x | x | AirPlay | o | x | x | x | x | x | x | x | x | x |
| -NDS | x | x | x | x | x | x | x | x | x | x | x | x | x |
| Desktop | | | | | | | | | | | | | |
| -Web with plugin | x | o | o | x | o | x | x | x | x | x | x | x | x |
| -Windows | o | o | o | x | o | o | x | o | o | x | o | x | x |
| -Mac OSX | x | o | o | x | x | o | o | o | x | x | o | x | x |
| -Linux | x | x | x | x | x | x | x | x | x | x | o | x | x |
| Console | | | | | | | | | | | | | |
| -PS3 | x | o | x | x | x | x | x | x | o | x | x | x | x |
| -Wii | x | x | o | x | x | x | x | x | x | x | x | x | x |
| -XBOX | x | o | x | x | x | x | x | x | x | x | x | x | x |
| **5. Price** | | | | | | | | | | | | | |
| Price | $99/seat + 25% royalty after $50000 of income | $ 3000 for iOS Pro | 1499 Euro | $2500 per seat per year | $ 500 | $500 per 1 application | 399.99 $ for iOS and 399.99 $ for Android | Open source | $ 4950 | $ 300 | Open source | Open source | Open source |

**Figure 4-26 Comparison Matrix**

Green color and red color show big advantage and big disadvantage of game engine in that criterion. For example, there are only UDK and Unigine game engine that has static global illumination, so they are marked with green color. Raydium, oolongengine, and jPCT do not have many tools, so they are marked with red color.

## 4.6 Conclusion

Comparison matrix shows that there is no best game engine for all aspect. Each game engine has their advantages and disadvantages. We separate winner game engine to two main categories. First, choosing game engine by the best game engine for each main aspect. Second, clarify the best game engine for each game genre.

### 4.6.1 Category by main aspects

As discussed before, the main aspect of 3D game engines are consist of programming libraries, usability, deployment platform, and price. The most important programming library is 3D graphics, this is only requiring library for 3D game engine. Usability divides into ease of learning and efficiency of use. As a result, 3D graphics, other programming libraries, ease of learning, efficiency of use, deployment platform, and price are the main evaluation aspect of 3D game engine.

#### *4.6.1.1 3D graphics*

UDK is the winner of this aspect. This game engine has all listed 3D graphics features in comparison matrix. This game engine makes the most beautiful graphics games in mobile devices, because UDK can bake global illumination to texture.

#### *4.6.1.2 Other programming libraries except 3D graphics*

Winner of this aspect also goes to UDK, it uses one of the best programming libraries for each criteria such as PhysX, fmod, and Bink.

#### *4.6.1.3 Ease of learning*

The winner of this aspect is Unity3D, because it has the best communities. Tutorial and examples from the game engine cannot compare to large communities. Communities are growing extremely fast compared to tutorial and examples that are provided by engine's creator. These communities made many code snippets and third party libraries. These extra code and framework really helps people learn the platform much faster. Moreover, since there are many people in many communities, some people will help answer question for beginner. Furthermore, Unity3D has asset store, which add even more tutorials and examples. Beginner can learn Unity3D platform fastest, because of these reasons.

#### *4.6.1.4 Efficiency of use*

There are many 3D game engines that can be the best of this aspect. UDK which is supports most of the criteria cannot be winner, because it doesn't have profiler. Profiler is very important for analyze and optimize the game. There are many candidates for this aspect. They also have their advantages and disadvantages. Therefore, not one of all game engines are the best for this aspect.

#### *4.6.1.5 Deployment platform*

Obviously, Shiva3D is the best in this aspect. Shiva3D's game can be deployed in more than ten platforms.

#### *4.6.1.6 Price*

The best game engine for "price" aspect is open source game engine. There are four game engines in comparison matrix, which are Ogre3D, Raydium, oolongengine, and jPCT.

# 5 Case Study

## 5.1 Demo Requirements

First, we receive goal of the app. The company wants 3D demo application on iOS and Android, which is capable to show features about airline search. They want application with high quality graphics and intuitive, because they want to impress their client. This project should be delivered in 2 weeks.

After we receive requirements, we design this application as 3D globe. This globe shows location of countries and airports around the world. Then user can select each airport to see information of each airport, which includes name, images, prices, and path to other airports. User can also filter air tickets by price and destination.

## 5.2 Choose 3d Game Engine from Comparison Matrix

Some criteria in comparison matrix that match with requirements are deployment platform and 3D graphics. Company wants both iOS and Android, so we can remove all game engines that do not support both platforms. We have 5 game engines left include Unity3D, Shiva3D, Marmalade, Esenthel Engine, and SIO2. With these engines Unity3D is the best engine for 3D graphics because only Unity3D can apply static global illumination to the scene. Since we need to implement this application within 2 weeks, ease of learning criterion is very important. Two weeks is very short time. Unity3D is also the best game engine in this criterion. Therefore, we choose Unity3D to implement this demo application.

## 5.3 Result

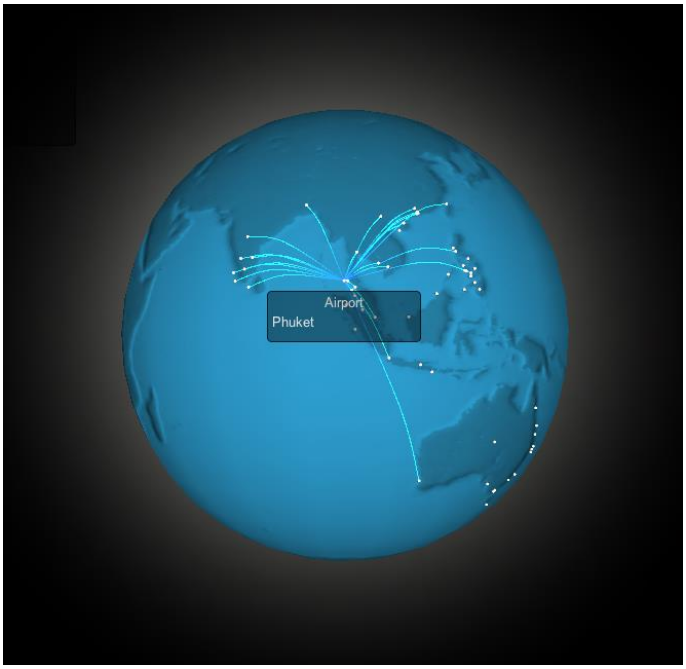


**Figure 5-1 globe application with selecting one airport**

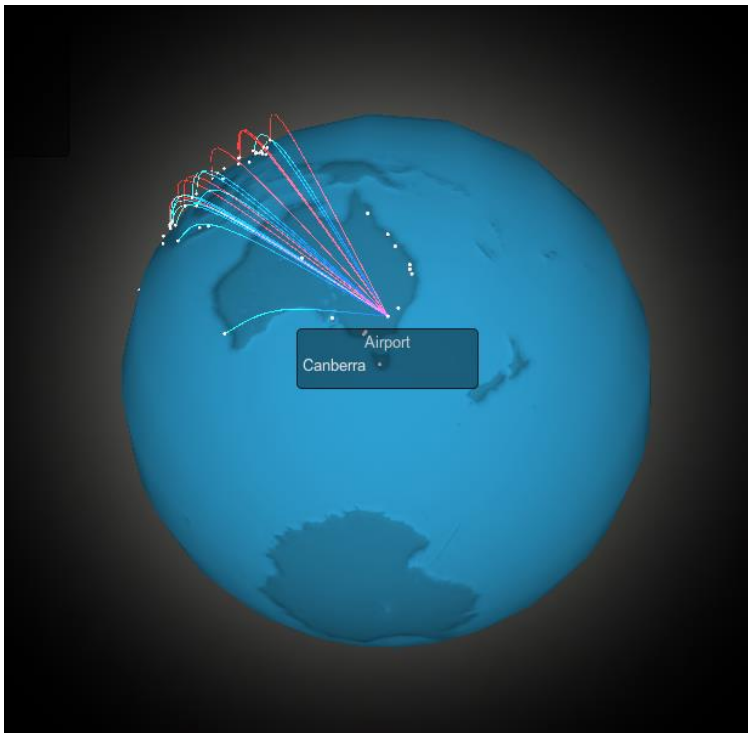


**Figure 5-2 Globe application with filtered path**


Figure 5-3 These two images show all travel routes for Tigerairways and current location


Figure 5-4 These images show how native UI works with Unity3D scene.


**Figure 5-5 These images show native map view that can show all Tigerairways's routes**

## 5.4 Feedback

This demo application is a utility application more than game application. So, User interface is very important. We underestimate graphical user interface (GUI) editor comparison criterion. Unity3D does not have user GUI editor. Moreover, it does not support GUI for touchscreen like scroll view. It has only scroll view for mouse input. We have to implement touch scroll view by ourselves. GUI in Unity3d doesn't automatically resize depends on screen size, so we have to scale it by ourselves. In addition, Zero point of GUI is on bottom-left, which is different from top-left for touch point. This is the reason why top selling product in assets store is GUI editor.


**Figure 5-6 NGUI is top selling product in Unity3D asset store.**

Since we do not have more budgets for purchase new component. It is much easier to use native user interface from iOS. We can use native user interface, because Unity3D can export to XCode project, which we can modify. There is no official support from Unity3D's owner. We manage to make it work by using plugin for Unity3D. Additionally, we can use all native UI such as map view, scrollable view, segmented view, tab bar, and etc.

3D graphics features of Unity3d are very easy to use. For example, Unity3D includes optimized mobile normal map diffuse shader in their editor. Furthermore, Unity3D editor can make normal map texture from image within editor. Unity3D takes few minutes to compile and run on actual device, but it has Unity remote application on device. So, game developer can see and try the application within few seconds. This application extremely improves workflow.

About ease of learning, as we think Unity3D is very easy to learn. There are so many examples, code snippet, and tutorials. First of all, there is one project in asset store that provides globe with map and geo location script [Figure 5-4]. We do not have extra money, and it does not very complicate. So we decide to implement it by ourselves. There are many articles about how to calculate position on sphere from latitude and longitude. They are not the best, because we still need to adjust it in Unity3D project. We found one article that include Unity3D project for free. Therefore, it is very easy to implement. This reminds me how important of large community.

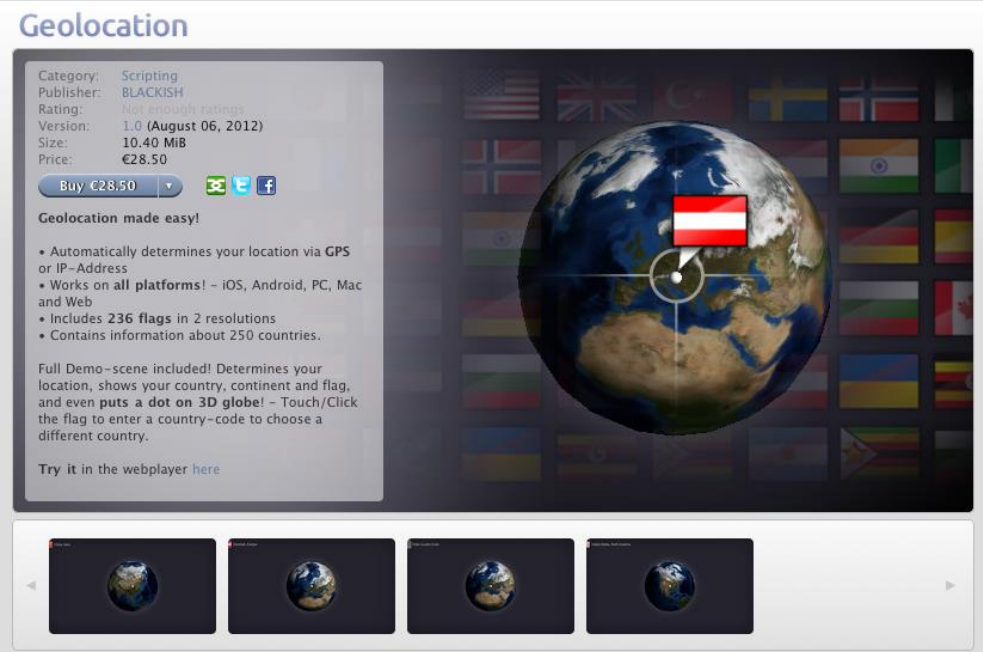


Figure 5-7 Geolocation package in Unity3D asset store

# 6 Conclusion

This thesis create comparison matrix for 3D mobile game engine. Since each game engine has many different features, we achieve our goal to help game developers pick 3D game engine that is the most suitable for them. This comparison matrix extracts correct features from entire game engines. Game developers can see the advantages and disadvantages of each 3D game engine very easily. Our case study shows that our chosen game engine has the same advantages and disadvantages information with comparison matrix information. For instance, Unity3D engine is bad at graphical user interface (GUI) in comparison matrix. After we tried Unity3D engine, we realized that GUI system is really bad in Unity3D engine.

We take most of the game engines in to account in this thesis. This comparison matrix shows general information of game engines. Those criteria are not specific information for some application.

## 6.1 Limitation

Since we have limited time, we cannot evaluate all game engines on the market. This thesis focuses on 3D and mobile game engine, so we exclude all 2D only game engine and non-mobile game engine.

Since game engine is software framework, their owners always improve their system. So, it is not possible to have only one comparison matrix for long time, but comparison criteria are still useful for future game engine.

### 6.2 Future Work

We realized one problem about our comparison matrix. Since we are making comparison matrix for mobile platform. We should change "GUI Editor" criterion to "Mobile GUI Editor" because there are so many different features between mobile GUI and desktop GUI. For example, user can only scroll the scrollable view with roller on mouse or click at scroll bar, but user can scroll the scrollable view by touch and drag on the actual content of the view. This means that scrollable view in desktop will not usable for mobile platform.

We need to summarize the result, so we can not have all details of every game engine. Otherwise comparison criteria will be very large; therefore it will be very hard to read. In the future, we can make search application for mobile game engine. This application will have search and filter features. These features help game developer to choose their best game engine easier. For example, developers can choose all game engines that have map editor, PhysX, and price less than 10000US dollar. Furthermore, this application can have and easy way to add information. We can provide update information for comparing game engine.

## 7 References


Zachte, E. (2012). *Wikimedia*. Retrieved from Wikimedia Traffic Analysis Report - Operating Systems: http://stats.wikimedia.org/archive/squid_reports/2012-08/SquidReportOperatingSystems.htm

WWDC Apple Design Awards. (2011). Retrieved from https://developer.apple.com: https://developer.apple.com/wwdc/ada/

Wikipedia. (2012). *List of game engines.* Retrieved from https://secure.wikimedia.org/wikipedia/en/wiki/List_of_game_engines

Wikipedia. (2011). *Physics engine.* Retrieved from https://secure.wikimedia.org/wikipedia/en/wiki/Physics_engine

wp7applist. (2011). *Apps Stats.* Retrieved from http://wp7applist.com/stats/

Xiaoting Wang, X. M. (2010). A Script-based 3D Game Engine—WGE. *Biomedical Engineering and Computer Science (ICBECS)* .

vgsales. (2008). *vgsales.wikia.com.* Retrieved from vgsales.wikia.com: http://vgsales.wikia.com/wiki/Video_game_industry

Abdullah Al Mazed, M. H. (2005). INCREASING PERFORMANCE OF 3D GAMES USING PRECALCULATION. *9th International Multitopic Conference, IEEE INMIC 2005* .

Adrian Boeing, T. B. (2007). Evaluation of real-time physics simulation systems. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* .
Arstechnica. (2007). *Arstechnica.* Retrieved 2007, from arstechnica.com: http://arstechnica.com/gaming/news/2007/08/gaming-to-surge-50-percent-in-four-years-possibly.ars
Blow, J. (2004). Game Development: Harder Than You Think. *Queue - Game Development* .
Eberly, D. H. (2006). *3D Game Engine Design, Second Edition: A Practical Approach to Real-Time Computer Graphics.* Morgan Kaufmann Publishers.
Epic Games. (2011). *Unreal Engine 3 Features.* Retrieved from http://unrealengine.com/features
Epic Games. (2011). *The Art of Infinity Blade.* Retrieved from www.unrealengine.com: http://www.unrealengine.com/files/downloads/GDC2011_IB_BlackMagic_Final.pdf
Dan Fu, R. J. (2008). Evaluating Game Technologies for Training. *Aerospace Conference, 2008 IEEE* .
devmaster.net. (2011). *DevMaster's Game and Graphics Engines Database.* Retrieved from http://www.devmaster.net/engines/
DevMaster.net. (2012). *DevMaster's Game and Graphics Engines Database.* Retrieved from http://www.devmaster.net/engines/
DesuraNET Pty Ltd. All Rights Reserved. (2012). *MODDB.* Retrieved from http://www.moddb.com/engines
Dieter Fritsch, M. K. (2004). Visualization Using Game Engines. *ISPRS* , 621-625.
DigitalBuzz. (2011). *Infographic: Mobile Gaming Statistics 2011 .* Retrieved from http://www.digitalbuzzblog.com/infographic-mobile-gaming-statistics-stats-2011/
Donald Mustard, G. M. (2011). *The Art of Infinity Blade.* Retrieved from www.unrealengine.com: http://www.unrealengine.com/files/downloads/GDC2011_IB_BlackMagic_Final.pdf
Frederick M. Weinhaus, V. D. (1997). Texture mapping 3D models of real-world scenes. *ACM Computing Surveys (CSUR)* .
Jakob Nielsen, H. (2006). *Prioritizing Web Usability.* New Riders Press, Berkeley CA .
Jakob Nielsen, H. (2006). *Prioritizing Web Usability.* New Riders.
John M . Airey, J. H. (1990). Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *I3D '90 Proceedings of the 1990 symposium on Interactive 3D graphics* , 41-50.
John Rohlf, J. H. (1994). IRIS performer: a high performance multiprocessing toolkit for real-time 3D graphics. *SIGGRAPH '94 Proceedings of the 21st annual conference on Computer graphics and interactive techniques* .
MACEDONIA, M. (2002). Games Soldiers Play. *Spectrum, IEEE , 39* (3), 32 - 37 .
Miao Song, P. G. (2008). A framework for dynamic deformation of uniform elastic two-layer 2D and 3D objects in OpenGL. *C3S2E '08 Proceedings of the 2008 C3S2E conference* .
Miao Song, P. G. (2008). A framework for dynamic deformation of uniform elastic two-layer 2D and 3D objects in OpenGL. *C3S2E '08 Proceedings of the 2008 C3S2E conference* .

Ming-Hsin Tsai, C.-H. H.-Y. (2006). Game Programming Courses for Non Programmers. *Proceedings of the 2006 international conference on Game research and development* .
Mohd Fairuz Shiratuddin, D. F. (2007). UTILIZING 3D GAMES DEVELOPMENT TOOL FOR ARCHITECTURAL DESIGN IN A VIRTUAL ENVIRONMENT. *7th International Conference on Construction Applications of Virtual Reality.*
Panagiotis Petridis, I. D. (2010). An Engine Selection Methodology for High Fidelity Serious Games. *VS-GAMES '10 Proceedings of the 2010 Second International Conference on Games and Virtual Worlds for Serious Applications* , 27-34.
Peter Lindstrom, D. K. (1996). Real-time, continuous level of detail rendering of height fields. *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* , 109-118.
Schramm, M. (2008). *Chinese WoW hits 1 million concurrent players.* Retrieved from http://wow.joystiq.com/2008/04/11/chinese-wow-hits-1-million-concurrent-players/
Seung Seok Noh, S. D. (2006). Using a game engine technique to produce 3D Entertainment contents. *ICAT '06 Proceedings of the 16th International Conference on Artificial Reality and Telexistence--Workshops* .
StatCounter. (2011). *Global Stats Top 8 mobile OS.* Retrieved from http://gs.statcounter.com/#mobile_os-ww-monthly-200812-201109
Stefan Marks, J. W. (2007). Evaluation of Game Engines for Simulated Surgical Training. *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* .
Techcrunch. (2011). *Nokia Confirms Microsoft Partnership, New Leadership Team, Organizational Changes.* Retrieved from http://techcrunch.com/2011/02/10/nokia-confirms-microsoft-partnership-new-leadership-team/
Terdiman, D. (2011). *New technology revs up Pixar's 'Cars 2'.* Retrieved from CNet: http://news.cnet.com/8301-13772_3-20068109-52/new-technology-revs-up-pixars-cars-2/
Tobias Ritschel, T. G.-P. (2009). Approximating Dynamic Global Illumination in Image Space. *I3D '09 Proceedings of the 2009 symposium on Interactive 3D graphics and games* .
Tom True, B. G. (2004). Performance OpenGL: Platform Independent Techniques. *SIGGRAPH '04 ACM SIGGRAPH 2004* .
Tschida, C. (2010). *The Unreal Engine Can Do For iOS Gaming With Epic Citadel.* Retrieved from appadvice: http://appadvice.com/appnn/2010/09/epic-games-shows-unreal-engine-ios-epic-citadel