COMPARISON OF GAME ENGINES

NOPPAWAN CHUMNANCHANG

PARAWADEE KHRAICHOM

A SPECIAL PROJECT SUMMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE (INTERNATIONAL PROGRAM)

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACEDEMIC YEAR 2011

Thesis Title	Comparison of Game Engines	
Student	Ms. Noppawan	Chumnanchang
	Ms. Parawadee	Khraichom
Degree	Bachelor of Science	
Program	Computer Science	
Year	2011	
Thesis Advisor	Assoc. Prof. Dr. Veera Boonjing	

ABSTRACT

In the present, there are many game engines for game developers to create game project. Some game engines are easy to be used and learned, some game engines have more assets with editor while some game engines have less assets to create game projects. This special project demonstrates the comparison of game engines and evaluate with criterion for the beginning game developers to select the game engines to be used with creating game project easily. Nowadays there are many game engines advertised in the Internet and some people post review about the game engines on web board which make game developers have several choice of game engines to choose. Therefore, the objective of this project is to compare more game engines which are GameCore, Unity, and CryEngine and provide the evaluation for the game developers beginners.

Keywords: Criterion, Evaluation, Game developers, Game engines.

ACKNOWLEDGEMENT

This special project is the comparison of game engines that cannot be successful without help from some persons. In this opportunity, team of project creation would like to thank you for their kindness as the following,

- *Our parent* who foster, teach, support in education, give spirit, and help us everything.

- *Assoc. Prof. Dr. Veera Boonjing* who is our good consultant and supervisor to give us the topic of this special project.

- *Assoc. Prof. Dr. Jeeraporn Werapun and Dr. Rungrat Wiangsripanawan* who are our committee to comment and help us to solve the problem for our special project.

- *Our friends in King Mongkut's Institute of Technology Ladkrabang* who give spirit, help, and editing guideline when we have some problem.

In addition, we would like to take this opportunity to thank all people who help, consult, suggest and give spirit for this special project.

Noppawan Chumnanchang

Parawadee Khraichom

Table of Contents

Table of Contents (Cont.)

Table of Contents (Cont.)

Table of Contents (Cont.)

Table of Contents (Cont.)

List of Figures

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

List of Tables

Chapter 1

Introduction

1.1 Rational and Research Motivation

Nowadays there are many game engines in the world. Each game engine has the difference in advantages and disadvantages which make game developers confused to choose which one is an appropriate to be used. Game developers should have an evaluation to compare each game engine prior to create game. Each evaluation should cover about editing, content, and game play of each game engine.

Three game engines as GameCore, Unity and CryEngine were chosen to compare among of them in this special project. The reasons to choose these three game engines are as follow:

- GameCore is a familiar game which we have ever studied in class.
- Unity and CryEngine are well-known game engines for game developers.

1.2 Objective

The objective of this project is to compare GameCore, Unity, and CryEngine including the evaluation for game developers to choose game engines to create games.

1.3 Scope

This project will demonstrate how to compare GameCore, Unity, and CryEngine. The criterion used for evaluation are listed:

1.3.1 Compare game engines by evaluation for game developers

1.3.2 Criterion to create an evaluation which compose of the following,

- Editing
- Graphic
- Platform
- Content
- Game play

1.3.3 Software

- GameCore
- Unity
- CryEngine

1.3.4 Hardware

- The high performance in graphic personal computer or notebook

1.4Benefits

This special project will provide the following benefits for game developers to find the most appropriate game engines to create a game.

1.4.1 To help game developer to choose game engines easily.

1.4.2 To help game developers to choose game engines according to requirements.

1.4.3 To provide knowledge and evaluation method for game developers to select a suitable and good game engines.

1.4.4 To provide the comparison method of each game engines for game developers.

1.5 Organization

This special project consists of 5 chapters.

Chapter 1: Introduction

Chapter 2: Background of the special project

Chapter 3: Program work

Chapter 4: Experiment of this special project

Chapter 5: Conclusion of this special project

Chapter 2

Background of Game Engine

A game engine is a designed system for creation and development of video games. Nowadays, there are many Game Engines such as Unity3D, Game Core, Delta3D, Blender were created and developed for game developers. In order to reduce time for game creation, game developers should choose the appropriate game engine to create their games.

2.1 The Basic Process of Game Engine

The main functions of game engines is game creation time reduction and provide the easier method for game developers to develop their games. In the past, game developers had to write the initial program to command their game but right now games are more complexity which will make game developers take long time to create their games. Thus, game engines are the best choice for game developers. The property of each game engine is different such as, SCUMM is aim to create adventure 2D games, Unreal engine to create First-Person Shooting (FPS) Games. Game developers should choose game engines which have similar function to their games and modify to their own game creation.

2.2 Game Engine Design and Functional of a Game Engine

A game engine is a complex software system which is necessary for game developing and playing. Two different games with the same underlying engine can be difference in the game content such as graphics, sounds, storyline and etc. Game engines build a bridge between this content and the underlying hardware. With the help of an operating system abstraction layer, the same game content can be run on many platforms (e.g. Windows, Linux, XBox) without change.

The modern game engines consist of all or a subset of functional blocks depicted in figure 2.1.

2.2.1 Graphic Engine

The graphics engine loads, displays, manipulates and manages all the data related to graphical content and visual effects. 3D models of objects, landscapes, buildings, objects,

animals, and players can be loaded, textured, lit, and animated. Additional effects (e.g. blurring, lens distortion, depth of field) can be added to enhance the visual realism. Particle systems are utilized to simulate file, smoke, bubbles, blood, etc.

2.2.2 Audio Engine

All of audible content such as sound effects, ambient noise, and music is handled by the audio engine. To connect with modern soundcards, it is possible to simulate acoustic obstruction by objects, environments other than air, reverb, Doppler Effect and the spatial position of sound sources.

2.2.3 Memory and Process Management

The total memory usage of game content is often higher than the memory provided by the gaming platform. Because of all of this data is not needed simultaneously, the Memory Management is responsible for purging unused content from memory and in turn pro-viding and managing requested memory for new content. Modern engines parallelize tasks like graphics, sound, physics, and AI. To balance the workload of all these tasks efficiently, especially on multiprocessor platforms, the Process Management is utilized.

2.2.4 Event Handing

Another essential part is the Event handling. Input devices, like joysticks, mice, keyboards, and gamepads generate events as well as network, timers, other components of the gaming hardware, game scripts and many other sources. These events are handled, filtered and distributed by one central event loop.

2.2.5 Streaming

Some media such as music or video are not required to be loaded into memory before being played back, but can be instead of streamed to save precious memory resources. The Streaming mechanism is also capable for loading resources via the network from other servers.

2.2.6 Physics Engine

The physics engine implements advanced mathematical models f or calculating rigid body simulations of arbitrarily shaped and articulated objects (e.g. vehicles, machines). With the

development of highly sophisticated physics engines like Havok Physics™ or PhysX™, the simulation of soft bodies, cloth, fluids, and smoke become to be possible, accelerated either by specialized hardware or the computational power of the graphics hardware.

2.2.7 Artificial Intelligence Engine

Artificial intelligence, provided by the AI Engine, is required for controlling Non-Player Characters (NPC), the computer controlled antagonists in games. NPCs have to make decisions about how to follow, avoid or ambush the player, and how to react to aggressive or defensive actions in a realistic and effective manner. AI Engines incorporate computer science topics like neural networks, state machines, A* search, and much more.

2.2.8 Scripting

The flexibility of game engines is their greatest strength in creating manifold content. This is achieved by scripting languages that allow an immediate access to the functions of the game engine. By scripts, the game content gets its typical "fingerprint", how a game is to be played and controlled, how the story develops, and how interactive and immersive the game environment appears.

2.2.9 Networking

The final important functional block of a game engine is networking. By sending the state, movement and other information of each player and NPC over the network, other connected human players can collaborate in a game, because they all see the identical state of the game world at the same moment. The networking functions cope with network problems like packet loss or different runtimes of data packets from clients to servers and vice versa.

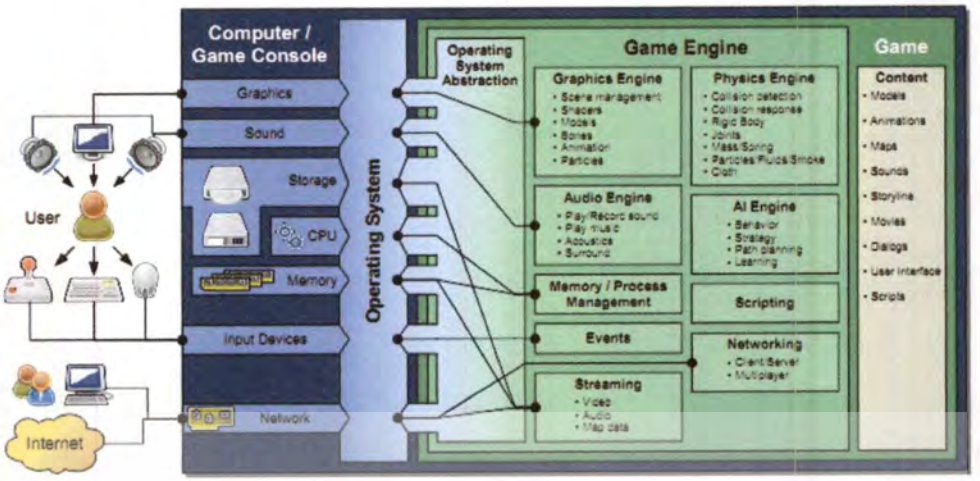

Figure 2.1: Functional Blocks of a Game Engine

2.3 Typical Game Engine Features [4]

Game engines can be evaluated according to their supported features. Below is a list of game engine features which are typical for modern day engines. Some of the features are only relevant to 3D engines, but most are typical in any kind of game engines. Complete 3D game engines with built-in editors are complex software programs. To get a general view of features for modern 3D engines.[2]

2.3.1 AI (Artificial Intelligence)

AI is short for Artificial Intelligence and it is used in games to give player an opponent to play against. Also simple puzzle games like Bejeweled or Tetris have some kind of AI which decides what kind of pieces are set on the board or what kind of piece falls next. In the other hand AI can consist of multiple computer controlled soldiers working together as a squad or multiple squads on the game's battlefield

Advanced AI behavior describes what kind of actions AI can do in the game. Can the AI controlled characters use triggers, how they use different inventory items, how they choose their weapons?

Bot is a computer controlled "player" used in multiplayer games. Bots can be used both as enemies or team mates in the different kinds of game modes to stand in place of missing human players. Bots can be also used in offline matches for training.

Formations are related to Bots, can they work as a team and use formations? Or can player give orders to the Bots?

Fuzzy logic replaces the boolean logic (true or false) with multiple degrees of the truth, vague definitions of different sets. This can be used in the game's AI code.

Line of Sight (LOS) checking is used to determine if a one can see from point A to point B inside a game level. Bots use these checks to determine if they "see" different objects on the level.

Natural language processing is used in some games so that player types in natural language sentences instead of choosing them from predetermined list. Check out a free game named Façade for reference.

Path finding is important as it is used by bots to find different places in the level. Some games like UT2004 use static path finding so that path nodes and waypoints must be placed to the map for the bots to work. The other games use dynamic style path finding so that the bots learn the levels during playing.

Scripting support determines whether bots in the game can be scripted to do certain sequences. This can be used to make cut scenes for example.

State machines are used when programming AI to the game. For example UT2004 uses states in it is bot code and there are different states for different behaviors like roaming, driving vehicles and fighting with the enemies.

Team based AI means that computer opponents or player's allies can work as a team making decisions based on the status of the whole team.

2.3.2 Animation

Animation is used in many different elements and it can be either 2D or 3D. 2D animation can be based on vectors or it can be traditional animation where each frame is drawn by hand. 3D animation is used to animate 3D models which can be player characters, space ships, flying birds or grass slowly moving in the wind.

Animation channels are the number of animation channels. One model can be used in the game.

Blending animations means that multiple animations are played simultaneously in the same model, blending the different animations together.

Facial animations are used to show emotions in game's characters. These can be different expressions like happiness, sadness, etc. See Half-Life 2 game for examples of good facial animation technology.

Exporters are used to convert models and animations from 3D modeling programs (Max, Maya, Blender, etc) into the game engine's format.

Inverse kinematics is used to determine the parameters of joints in order to go to the desired pose.

Morphing means the image changing from one to another with a transition.

Skeletal based animation uses a hierarchy bones to move the model and animate it.

Key frames are frames that are drawn or constructed by the animator.

Vertex (or mesh) based animation uses vertex positions to store the animation sequence.

Tweening (animation or shape) means that computer fills the gap between key frames.

2.3.3 Capacity

Engines have different kind of demands from the hardware and the following figures are used to describe the limits or power of the engines.

Frames per second (FPS) means how many frames in a second that the engine draw the game to the screen.

Number of network clients determines how many clients the engine support in multiplayer games. Typical FPS shooters have support for 4-64 players.

Number of polygons on the screen describes the amount of polygons on the screen. There are other factors like the use of textures which affects this number so it is hard to compare engines with this feature.

Number of objects on the map

Number of executed script commands per second, how fast is the scripting engine?

Size of textures, what kind of textures and how big does the engine support?

2.3.4 Cost and Licensing Terms

Using the other people's engines usually (but not always) costs money. To buy an engine from some developer, costs might include:

- Flat licensing fee which is paid when the engine is licensed from the engine developer
- Royalties based on number of copies sold
- Payments for additional platforms (Linux, PS2, XBox, etc)

Freeware engines are usually distributed in accordance with different license types. The typical licenses are GPL and LPGL. To put it simple, GPL can't be used at all with commercial applications because you have to release the source of the application. One way to take care of this issue is to try to negotiate buying a commercial license from the maker of the GPL'ed software.

GPL (General Public License) is commonly used license agreement with free software and is also used with some of the game engines. GPL is a free software license and allows people to use and or redistribute the software without being required to pay anyone. This license grants the following rights to the software user:

- Freedom to run the program, for any purpose.
- Freedom to study how the program works, and modify it. (Access to the source code is a precondition for this)
- Freedom to redistribute copies
- Freedom to improve the program, and release the improvements to the public. (Access to the source code is a precondition for this)

The primary difference between the GPL and more "permissive" free software licenses such as the BSD License is that the GPL seeks to ensure that the above freedoms are preserved in

copies and in derivative works. It does this using a legal mechanism known as copy left, invented by Stallman, which requires derivative works of GPL-licensed programs to also be licensed under the GPL. In contrast, BSD-style licenses allow for derivative works to be redistributed as proprietary software.

LPGL (GNU Lesser General Public License) is a free software license which is a compromise between GPL and simple permissive licenses. The main difference between the GPL and LGPL is that the LPGL can be linked to a non-(L) GPL program, which may be free software or proprietary software (which has restrictions on using and copying it).

The LGPL places copy left restrictions on the program itself but does not apply these restrictions to other software that merely links with the program. There are, however, certain other restrictions on this software. Essentially, it must be possible for the software to be linked with a newer version of the LGPL-covered program. The most commonly used method for doing so is to use "a suitable shared library mechanism for linking". Alternatively, static linking is allowed if either source code or linkable object files are provided.

2.3.5 Debugging

Engines are complicated systems so usually they have many debugging systems to help developers find the problems and fix the bugs from their code. The simplest engines have just logging but most advanced can have profilers, run time debugging and even options for simulating different kinds of network environments.

Logging is used to write things to the log file from the code.

Profilers can be used to find the bottlenecks from the code, places which take a lot of time to execute and which should be optimized.

Run time debugging means that program code can be traced during it is executing, going through the code lines one by one.

Simulating different kind of network environments like lag and latency is beneficial for testing the game in all kinds of environments.

2.3.6 Documentation & Help Channels

Because game engines are complicated, good documentation is needed so that developers can use them. Descriptions of typical documentation and help channels are below.

Books are about the game engine or developing games with it.

Developer forums, news groups and mailing lists are places where the game engine developers and people using the engine meet to discuss their problems, feature requests, etc.

Email to developers or calling them are the most direct ways to get help. It is another matter if the developers respond to your mails and calls.

Level of documentation describes how thoroughly the engine is documented.

Online help is usually located on the game engine developer's website and is usually more up-to-date when compared to offline help files.

Sample code / scripts / games help game developers to know the engine. It's much easier to study sample code than to start from scratch.

Tutorials (text, image based, flash, video) in addition to sample code are a good way to learn more about the used engine.

2.3.7 Graphics API

Different engines use different graphics APIs which provide primitive functions used to build a graphics library.

DirectX (or actually direct 3D) is a graphics API used only in Microsoft operating systems.

OpenGL is an open graphics API and can be used with many operating systems.

Software rendering means that the graphics are drawn by the CPU instead of the 3D graphics card. This makes it much slower than DirectX or OpenGL.

2.3.8 Hardware Requirements

What kind of hardware does the engine need in order to function properly? These are usually referred as hardware requirements.

3D card is used to draw graphics to the screen. Different cards support different DirectX versions and have different memory sizes.

CPU speed is reported in GHz, like for example 2 GHz CPU or better.

DVD-Drive (or CD drive) so that the game can be installed.

Hard drive space, how much space does the engine take? What about other game's assets like maps, sounds, textures, models, etc?

Internet connection (modem / ADSL) to be able to play the game in Internet.

Memory requirements differ between engines. This is usually reported in megabytes (MB) and the requirements can be anything from 16 megs to 512 megs.

Sound card is usually a requirement for playing games but the model is not important at the present.

2.3.9 Library vs. Editing Tools

This game engine feature describes what kind of support the engine has. The engine might be just a collection of all files (libraries) which hold C++ code. The libraries could be used just for drawing models on the screen and nothing more. On the other side the engine might include a comprehensive editor which is used to make levels, import art & sound assets, code the game logic, build menus, animate characters, etc.

Source code does the engine come with source or not? The code can be anything from C++ to Java.

Library files, does the engine consist of just library files (dll) so that you have to make the executable yourself?

Script editor, you can write code for game with this editor instead of using other program like notepad.

Map editor is used to make maps for the game. Some map editors use to build the entire level from BSP geometry while other map editors use to place 3D models and other objects to the map.

Path editor is used to place waypoints or path nodes to the map. Some path editors also include auto path option.

Terrain editor is used to build terrain for the map.

Sound and music editor is used to import sounds and music to the game.

Cut scene editor is used to make in game cut scenes which are also known as Machinima.

Particle editor can be used to create particle effects for the game.

Physics modeling tool can be used to define different kinds of physics attributes to the models and objects of the game.

Textures & material editor allows the developer to add textures to the game define their properties and make different kinds of shedders.

Animation editor is used to import animations made in 3D modeling programs and tweak them to be used in the game.

Built-in editor combining different editors into one package is the most ideal choice. Game engines like Unreal or Source have the most complete built-in editors.

2.3.10 Lighting

Maps in the game can have different kinds of lighting. The lighting is important part of the game's look.

Anisotropic filtering enhances image quality on surfaces which are steeply angled or far away.

Dynamic lighting means that the lights can change in the game, like for example player can use a torch to light up a room.

High dynamic range (HDR) lighting or rendering is a technique to get authentic looking lighting to the game. It basically adds more possibilities to the contrast values of the rendered image.

Light effects like blinking, wavering, pulse, disco, etc.

Light maps are used to store the brightness of map's surfaces.

Per pixel, calculating light for each pixel in the rendered image. This produces more realistic images than per vertex lighting.

Per vertex, calculating light for each vertex.

Radiosity is a lighting mechanic which simulates many reflections of light in the map. This means usually natural shadows and reflections.

Static lighting, the whole map or a zone is lighted with one static value.

Shadows, is produced by different rendering techniques. These can be also tricked in some engines by using projectors.

Volumetric, light rays are visible because of mist, fog, etc.

2.3.11 Localization

Since games are made for global markets, it is important that they should support localization. The things that need localizing in the game are Text and speech sounds.

2.3.12 Materials & Textures

Textures are used in many kinds of things from Hud graphics to 3D model skins.

16/32/64 bit textures, have different amount of colors.

Antialiasing is used to minimize jagged edges.

Effects with materials include for example bloom, depth-of-field, motion blur and halo

Material system includes for example technologies like alpha channels and bump maps.

Mipmaps are smaller versions of the original texture and their use is based on the distance of the player from the texture. The idea is to speed up rendering.

Multi textures mean that one model can have multiple textures.

Procedural textures are generated by algorithms.

Size limits, what kind of textures can be used? Some engines use textures which must be sized with power of 2 (256x256, 512x512, 1024x1024, etc).

Supported formats, what formats does the engine support? (bmp, jpg, png, dds) The format usually determines what kind of compression the images have.

2.3.13 Network

Game engines with network support provide different kinds of features described below.

Streaming means download content from the server on the file.

Player count is the number of players the game can have in a game. This can be anything from 2 to hundreds.

Level of network implementation, on what level is the network functionality and implemented for example script coding can be used to extend the network functionality.

Network games between different platforms can PC and console players play on the same server in the Internet?

Massive multiplayer games might have hundreds or thousands of players playing at the same time.

Performance, how much bandwidth is needed for the server or clients. Also what kind of ping is needed to be able to play the game?

Protocol, TCP / IP or UDP

Security, how secure is the network game against cheating?

Support for game play features describes if features like physics are supported in network games.

Server browser contains a list of servers on the internet.

Stat system, is there functionality ready for keeping player stats?

Type of network, different models include for example client / server, peer-to-peer and Master Server.

2.3.14 Other Features

Game engines usually have some of the following specific systems.

Demo record & playback system is used for recording offline or online games to a file which can be played back later.

In-game cut scenes (also known as machinima) are used to make and playback cut scenes made with the game engine.

Instant replay system for game events is important especially in sports games where recent game events can be played back.

Level of detail system (LOD) is an optimization technique which is based on drawing models with fewer polygons when they are further away.

Load & save system is used to save the game in some state and later load that save game to continue playing from the saved position.

Video playback, the format might be avi or mpg for example.

2.3.15 Physics

During the recent years, game engines have started supporting physics simulation. While being a complex subject matter, the features described below offer some insight into what kinds of things engines support regarding physics.

Buoyancy is how objects behave in fluids in the game.

Collision detection is one of the most important features of physics. It used to determine collisions between objects. Usually engines use simplified collision models instead of using the model's actual polygons to determine collisions.

Flight model is used to describe the simulation of flight physics in flight sim games.

Fluid simulation, for example water.

Movement mode likes walking, falling, flying, swimming, gliding, walking on walls, etc.

Network support for physics describes whether the engine supports physics in network games. Some games have just client side physics so that rag dolls for example behave differently on different clients since they don't affect game play.

Physical properties like mass, friction, gravity and inertia. Also is related to the object properties and interactions with each other like colliding objects, falling, rolling, hinges, balls, sockets and constraints.

Projectiles, physics for grenades or even bullets for example, is important in FPS games.

Rag dolls are used when game characters die and physics are used to simulate the behavior of their bodies while they fall, fly through the air hitting objects or fall down the stairs for example.

Vehicle physics are used to simulate the functionality of different vehicles and driving them. Examples of vehicles are motorcycles, cars, hovercrafts, airplanes, etc.

2.3.16 Programming Language

What language can be used to make games for the game engine? Also what language has been used to write the engine itself? The most common language is C++ but also other languages like Java are used. Some engines have support for multiple programming languages ready or there might be some 3rd party programs or libraries which use the engine as their basis.

2.3.17 Required Asset File Formats, Special Tools, etc.

Some game engines support only a couple formats while other's support a wide variety of formats. Some engines require specifically the use of some 3D program to make their levels. Also the game source code might be designed to be compiled only with a certain compilers and it requires a lot of work to make it compile on some other environment or compiler.

2.3.18 Scripting

Instead of hard coding the functionality to the source code, game engines can support some scripting language which is usually used to program the game's functionality. It depends on the engine what things are hardcoded and what can be scripted. Game engine can have its own scripting language like Unity 3d that can use some common language like JAVA, C# or Python.

2.3.19 Sound & music

Sounds systems for game engines have been neglected in the past but modern engines are starting to use 3D effects and other features in their sound systems. Also other advanced audio features like VOIP support and TTS are used more and more.

3D sound positioning means that the sound can be placed on the level and when player hears the sound, it sounds like coming from the correct direction based on the positions.

Basic sound parameters, for example doppler shift, level, filtering, pitch, looping, etc.

Dolby digital is a digital coding method. It is normally associated with version 5.1 which contains five channels + low frequency subwoofer channel.

EAX is Sound Blaster's technology for creating more ambiences with 3D sound to games.

Music support, supported formats can be for example ogg, mp3, midi, etc.

Network support for sounds, help how the sounds work in network games.

Obstruction does the sound system take obstacles like walls between player and the sound source into account.

Specialization makes the sound appear to come from specific location in the game level.

Streaming, the sound is loaded and played at the same time.

Text to speech (TTS) uses a speech synthesizer to convert text into speech sound.

Voice over IP (VOIP) is technology used to enable voice conversations between people over the Internet.

2.3.20 Special effects

Game engines implement all sorts of special effects and most of these effects are visual.

Color correction, currently only a demo feature of Valve's source engine. A "filter" used to change the colors of the rendered scene.

Environment mapping is a method where cube mapping is used to make object look like it reflects the environment around it.

Fog can be based on distance or height for example. Distance fog is normally used to limit the drawing distance.

Lens flare is used to simulate the real world effect when camera is pointed toward a bright source of light.

Mirrors & reflections, the bouncing back of a light ray from a surface.

Particle systems are used to make special effects like explosions, fire, flowing water, sparks, dust, fire, etc.

Portals are used to draw an image from some part of the level camera into a texture.

Projector (also known as decals) are used to project a texture on top of the level geometry.

Skybox is used to make a distant background for a game level.

Weather effects like clouds, rain, snow and wind. The effects can use particle systems.

2.3.21 Supported Platforms and Operating Systems

What operating systems and platforms the game engine supports. Common platforms like Linux, Mac, Mobile Phones (S60, etc), Windows, WWW (HTML, DHTML, Flash, Shockwave, etc). Consoles are nowadays divided into handheld ones (Nintendo DS, Sony PSP) and normal ones (PS, PS2, Nintendo GameCube, XBox, XBox 360, etc).

2.3.22 Terrain

To create a game for game events happen outdoors, the game engine should support terrain. Terrain is the basically the ground of the level.

Basic terrain support, determines whether the engine support terrains.

Deco layers, generate Static Meshes everywhere on the terrain. For example grass which is automatically placed on the ground.

Deformable terrains determines if the shape of the terrain be changed runtime, for example adding holes to the ground because of explosions.

Size of terrain is the same as scene manger which can manipulate size.

Texturing, how is the terrain textured? Can the terrain use the multiple textures with blending?

2.3.23 Type of Engines

The kind of engines can be used to develop text adventure games using nothing more but text and sometimes images.

2D engines use 2d objects like graphics, numbers and texts to display information to the player about the game. Also other programs like Flash which can be used to create 2D games could be categorized as a 2D engine though its main purpose is for doing small applications for websites.

3D engines are the most commonly used game engines at the time of making this document. Usually players move in a 3D world looking the world either from the 1st perspective (like from the eyes of the game character) or from the 3rd perspective (behind the player seeing the back of the character on the screen). These kinds of engines are based on drawing polygons on the screen.

2.4 GameCore

2.4.1 GameCore Definition

GameCore; Creativity with the power of cutting edge 3D Game Engine Technology. Intuitive Editors for 3D Models, Terrain Creation, Special FX Creation, and setting Physics properties provide the ability to produce stunning 3D Games. Develop and Publish on either the Mac or PC Games or publish directly to the internet with your Web3D player in figure 2.2.

Figure 2.2: Example of game from GameCore

2.4.2 User interface of GameCore

2.4.2.1 First Launch

Once you start program you will get interface as shown in Figure 2.3:

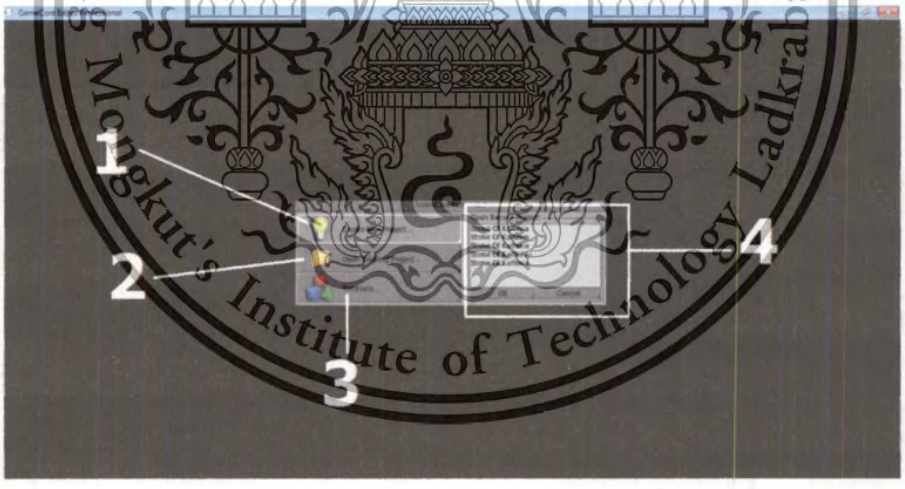

Figure 2.3: First Launch of GameCore

1. "Create New Project" - Click this command when you want to start new project.

2. "Open Existing Project" - Click this command when you want to edit your recent project.

3. "Tutorials" - Click this command when you want to learn how to use Game core.

4. In the box show the open recent project.

After you click "Create New Project" button you will get interface as shown in Figure 2.4 :

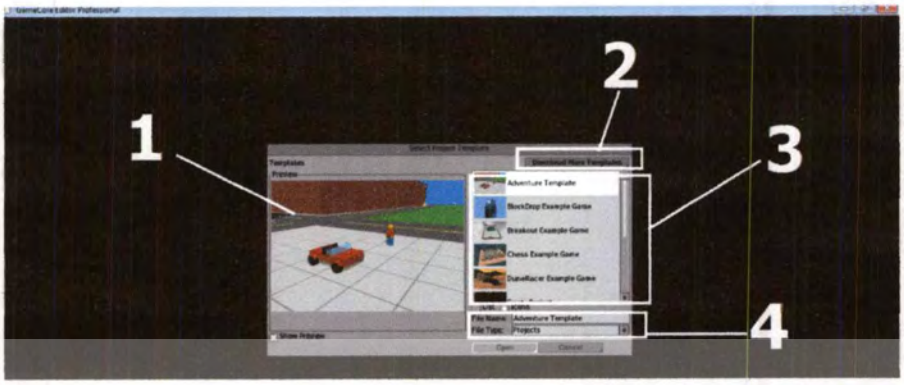

Figure 2.4: User interface that the GameCore is opened.

1. "Preview" - Show your select template.

2. "Download more templates" - Click this button to download templates form the website.

3. Show all templates that you have.

4. Show the file's name and file's type.

After you choose template you will get interface as shown in Figure 2.5 (In Figure 2.5 choose empty template).

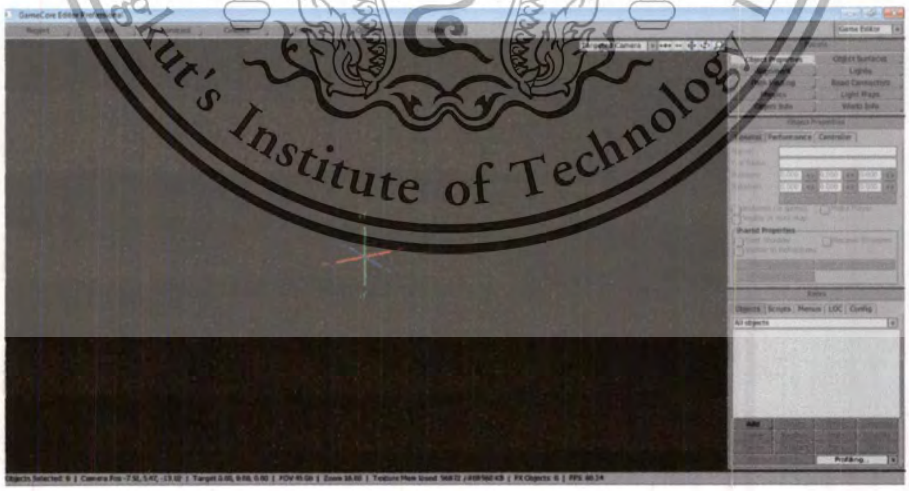

Figure 2.5: The GameCore is opened.

2.4.2.2 Toolbar

Figure 2.6: Toolbar of GameCore.

The toolbar is consisting of seven menus such as Project, Game, Environment, Cinema, Tools, Options, and Help (Figure 2.6).

a). Project

Figure 2.7: Project button of GameCore.

Project button include:

1. Open world - To open world in your project that you crate before.

2. New world - To create new world in project.

3. Save world - To Save the world in your project

4. Save World As - To save your world with new name.

5. Export Selected - To export the world in one project to another project

6. Import - To import the world from another project.

7. Open project - To open your project that you create before.

8. New project - To create new project.

9. Exit - To exit from program.

b). Game

Figure 2.8: Game button of GameCore.

Game button include:

1. Game resolution - You can choose the resolution by editor or game.

2. Startup World & Script - You can choose show splash screen or show launcher.

3. Run Game - Click this button to run game that you create.

4. Export Game - To export game to other files.

5. Export for Web - To export game for website.

6. Export File List - To export all file that you select to the websites or other file.

c). Environment

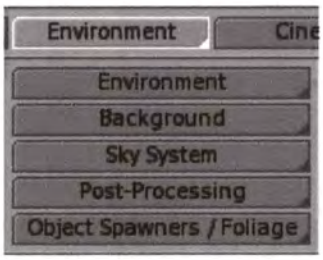

Figure 2.9: Environment button of GameCore.

Environment button include:

1. Environment - To add fog, sound and light.

2. Background - To add sky or game background.

3. Sky System - To edit objects in the sky such as cloud, sun.

4. Post-Processing - To edit color of game (R, G, and B).

5. Object Spawners / Foliage - To edit scale of the object.

d). Cinema

Figure 2.10: Cinema button of GameCore.

Cinema button include:

1. Show Cinema Panel - To show command of cinema player.

2. Load Keys - Load cinema form another file.

3. Save Keys - Save cinema.

4. Copy Camera Position - To capture picture form the cinema.

e). Tools

Figure 2.11: Tools button of GameCore.

Tools button include:

1. Font Builder - To edit text.
2. Export Cube Map - To export cube map.
3. Export Sphere Map - To export sphere map.
4. Export Geometry - To export geometry objects.
5. Create Mini-Map - To create the mini-map of the world.
6. Screenshot - To create and save screen shot.
7. Record Movie - To record the movie that you create on your world.

f). Options

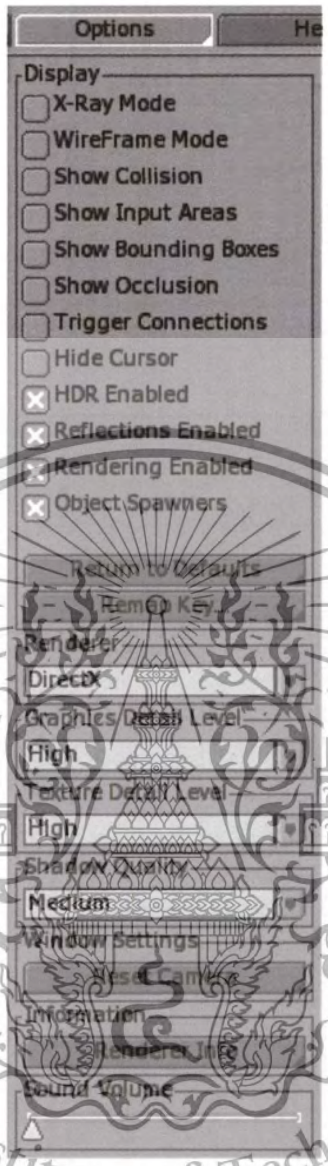

Figure 2.12: Options button of GameCore.

"Options" is about renderer process you can select what you want to see on world display **(such as X-Ray mode, input display, collision, cursor, bounding boxes and etc.) in figure 2.11.**

g). Help

Figure 2.13: Help button of GameCore.

"Help" is a function that helps you when you have the problem while you use Game core in Figure 2.13.

2.4.2.3 Control Tools

You can rotate your world by commands as shown in Figure 2.14.

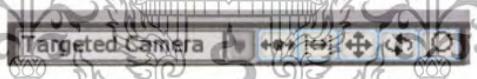

Figure 2.14: Control Tools of GameCore.

2.4.2.4 Layouts

The layouts are on the right side of GameCore and these three layouts have command to edit your game in Figure 2.15. The layouts consist of Panels, Object Properties, and Items.

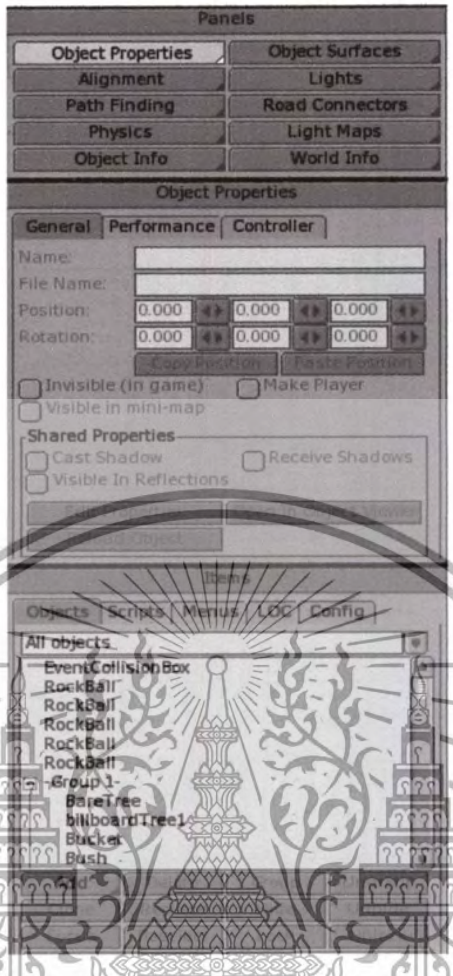

Figure 2.15: Layouts of GameCore.

a). Panels

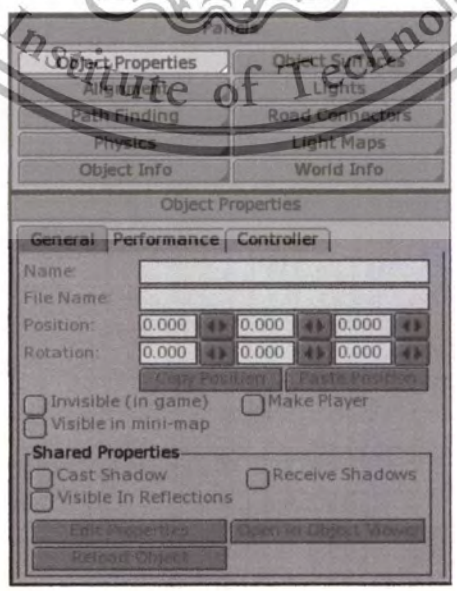

Figure 2.16: Panels of GameCore.

"Object Properties" - When you click the "Object Properties" the detail of object will show. You can change object's name, position and its properties (such as invisible in game, receive shadow, fade-in, fade-out and etc.) as shown in Figure 2.16.

"Object Surfaces" - You can change the objects surface (such as Smooth surface, vulgar surface, color, scale and etc.) as shown in Figure 2.16.

"Alignment" - Help you set the position of object on the terrain in the world as shown in Figure 2.16.

"Lights" - To create and add the detail of lights that you need in the world as shown in Figure 2.16.

Objects Info - Show detail of the object is selected in Figure 2.16.

World Info - Show detail of your world in Figure 2.16.

b). Item

Figure 2.17: Item of GameCore.

This box shows all objects that you add into your world. You can add, delete, group, ungroup, clone, replace, hide, unhide, focus, deselect, lock, unlock and rename group of all objects in figure 2.17.

2.5 Unity

2.5.1 Unity Definition

Unity is a game development tool that has been designed to let you focus in creating amazing games. Unity allows you to focus on simply creating for your game. Developed web or console is the tool for job as shown in figure 2.18.

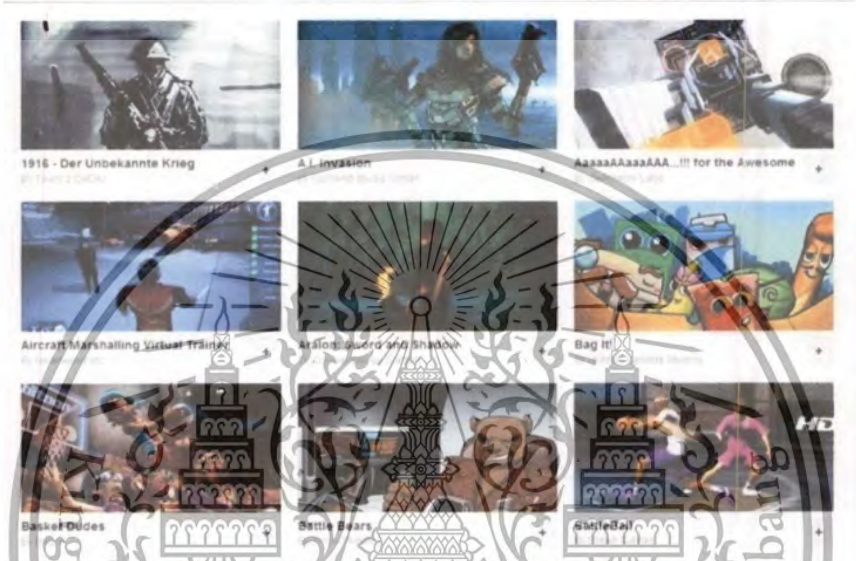

Figure 2.18: Game list from Unity.

2.5.1.1 Stunning Quality and Blazing Speed

Every game can be a masterpiece. Unity delivers the technology to impress and inspire your players with everything from jaw-dropping visuals to ambient sound scales. Even your browser game can look and feel like a state-of-the-art AAA title when it is powered by Unity. Your game needs speed and Unity defines fast. Unity is packed with new technology — ranging from geometry batching to occlusion culling — designed to squeeze every last bit of performance out of your next game. Regardless of your target platform, Unity will help you pull off as much as possible.

2.5.1.2 Intuitive and Powerful

At any time, simply hit play to test out your creation. The editor will emulate your target platform so what you see is what you publish. Use the built-in profiler to get full stats on all aspects of your game, from draw calls to audio to scripting performance. Assets are previewed in the editor, so you can quickly find what you're looking for. Tags and filters aid you in quickly finding what you need. The Unity editor is a fully featured world builder. Just drop your source

assets into the project and drag them into your scene. All public variables from scripts are exposed into the editor, so object references are easy to set up and game play can be tweaked instantly. Unity auto-unwrap light mapped models and lets you prioritize scene objects for aliasing. You keep working while Beast is baking so you're never held up.

2.5.2 User interface of Unity

2.5.2.1 First Launch

Game developers can open Unity by **Start->Programs->Unity** on Windows or **double click** on Unity icon on the desk top. The Unity will appear on your screen in Figure 2.19.

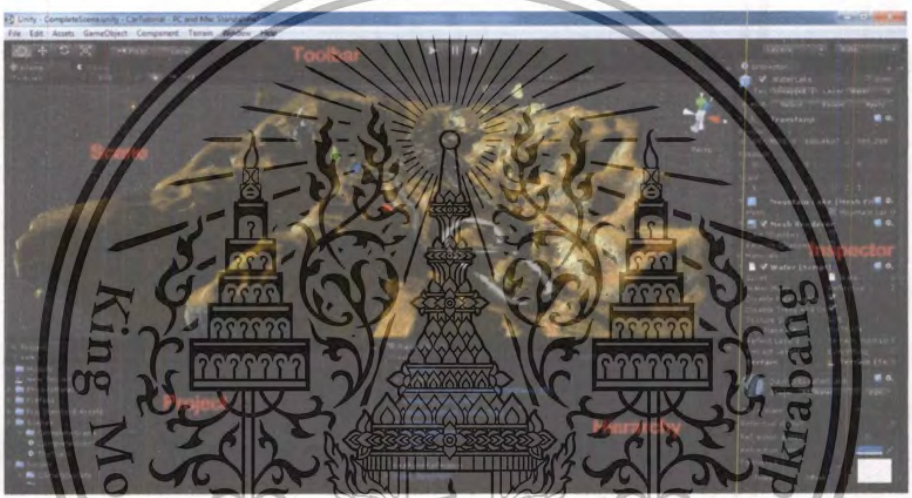

Figure 2.19: Unity 3D screen.

Game developers can manage custom workspace by them self. You can drop tap to other tap area for example, you can drop the project view to tap area next to Inspector in figure 2.20.

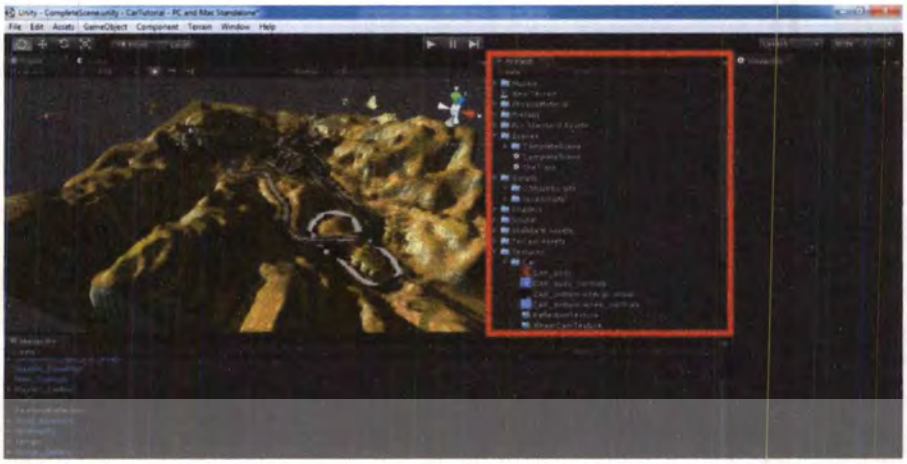

Figure 2.20: Management custom workspace of unity.

2.5.2.2 Components of Unity user interface

The Unity include many menus and GUI that have different functions for game developers to create game. There are toolbar, Scene, Project, Hierarchy, and Inspector. The explanation and detail of each menu are given below.

2.5.2.2.1 Toolbar

Figure 2.21: Toolbar of unity.

Toolbar consists of five basic controls and eight menus which each of them are related to the different parts of the Editor in figure 2.21.

Basic controls;

a). *Transform Tools*: Use with the Scene View. When we click "Hand icon" to drag the camera around (see Figure 2.22a). If we hold **Alt** and click "Hand icon" to drag orbit the camera around the current pivot point (see Figure 2.22b).

(a) (b)

Figure 2.22: Transform tools

b). *Transform Gizmo Toggles*: affect the Scene View display in Figure 2.23.

Figure 2.23: Transform Gizmo Toggles of unity.

c). *Play/Pause/Step Buttons*: Use with the Game View for play, pause step the game as example in Figure 2.24.

Figure 2.24: Play/Pause/Stop Buttons of unity.

d). *Layers Drop-down*: Controls which objects are displayed in Scene View in Figure 2.25.

Figure 2.25: Layers Drop-down of unity.

e). *Layout Drop-down*: Controls arrangement of all Views as in Figure 2.26.

Figure 2.26: Layout Drop-down of unity.

Menus;

a). File:

Figure 2.27: "File" menu of unity.

b). Edit:

Figure 2.28: "Edit" menu of unity.

c). Assets:

Figure 2.29: "Assets" menu of unity.

d). GameObject:

Figure 2.30: "GameObject" menu of unity.

e). Component:

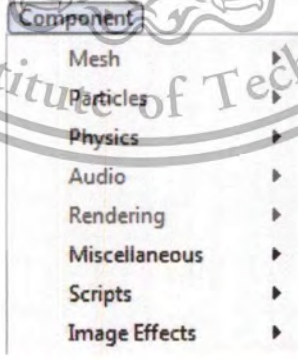

Figure 2.31: "Component" menu of unity.

f). Terrain:

Figure 2.32: "Terrain" menu of unity.

g). Window:

Figure 2.33: "Window" menu of unity.

h). Help:

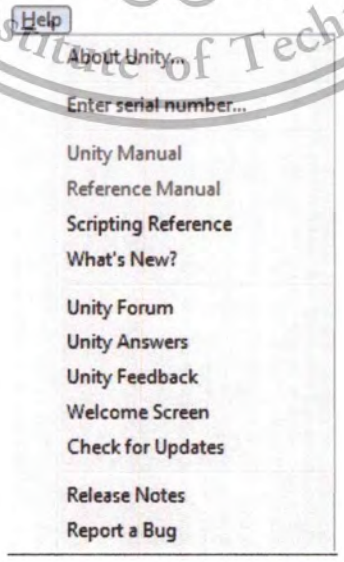

Figure 2.34: "Help" menu of unity.

2.5.2.2.2 Scene

The scene view is the interact sandbox. You can use Scene view to select the position in the scene view for creating game as shown in Figure 2.35.

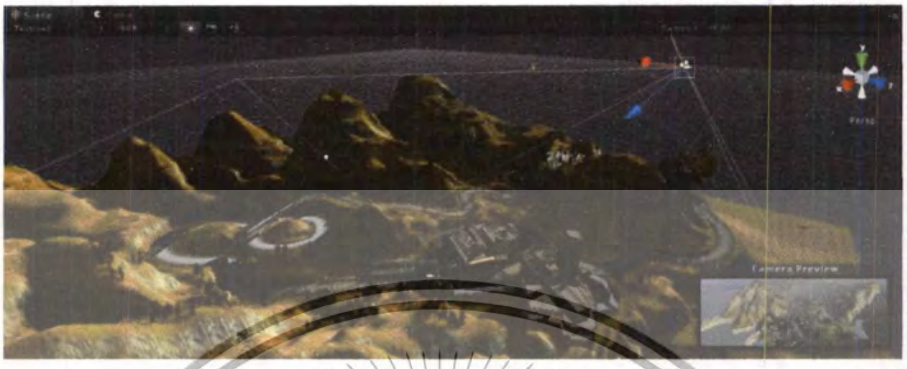

Figure 2.35: Scene view of unity.

In the scene view, you can drag camera around and drag orbit camera around the current pivot point. You can look to another side with x-axis, y-axis, and z-axis by Persp (Perspective) in Figure 2.36.

Figure 2.36: Persp (Perspective) for the scene view of unity.

When you click "x-axis" (see Figure 2.37), you can see the right of view (see figure 2.38).

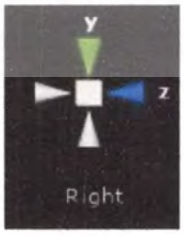

Figure 2.37: x-axis click.

Figure 2.38: The right of scene view of unity.

When you click "y-axis" (see Figure 2.39) ,you can see the top of view (see figure 2.40).

Figure 2.39: y-axis click.

Figure 2.40: The top of scene view of unity.

When you click "z-axis" (see Figure 2.41), you can see the front of view (see figure 2.42).

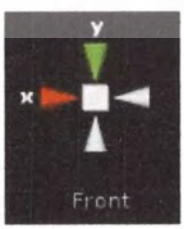

Figure 2.41: z-axis click.

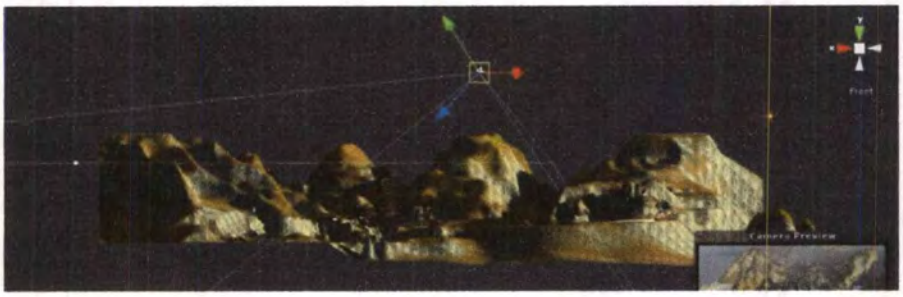

Figure 2.42: The front of scene view of unity.

You can see camera view of the game that is created before you want to run them when you click Game view in Figure 2.43.

Figure 2.43: The Game view of unity.

2.5.2.2.3 Project

In the Unity, a project contains an asset folder that stores all assets which we make up our game, like 3D models, sound, texture, scene, prefabs in Figure 2.44. You can import more assets to the project by drag any files from our computer to the project view or click on **Assets** menu and then click on **Import New Asset**. Our assets will be used in game.

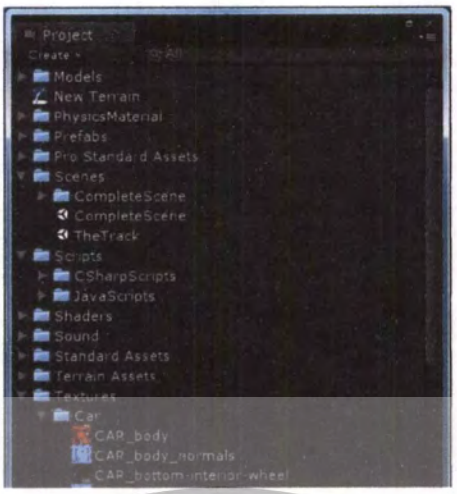

Figure 2.44: The project view of Unity.

2.5.2.2.4 Hierarchy

The hierarchy contains every GameObject in the current scene as shown in Figure 2.45. You can click on some object in the hierarchy and the selected one will be shown and then double click or press "F" key to focus the object that was selected. The hierarchy menu, can show or hide the parents of the object (see Figure 2.46)

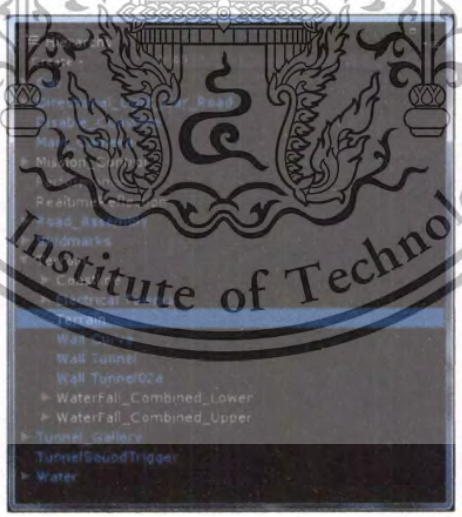

Figure 2.45: The hierarchy of Unity.

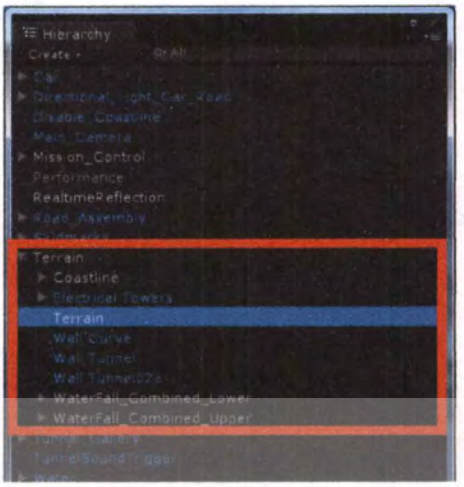

Figure 2.46: The parent hierarchy of Unity.

2.5.2.2.5 Inspector

The inspector shows detail of object or assets in game as in Figure 2.47. You can edit the detail of each object or assets and you can set the position of object, assets, or terrain in the inspector.

Figure 2.47: The inspector of Unity.

2.6 CryEngine

2.6.1 CryEngine Definition

The CryEngine is a highly advanced development solution that passes all expectations for the creation of blockbuster games, movies, high-quality simulations and interactive

applications. The third iteration of Crytek's proprietary CryEngine is the only all-in-one game development solution for the PC, Xbox 360 and PlayStation 3 that is truly next-gen ready (figure 2.48).

Figure 2.48: Game development solution

2.6.1.1 Real-Time Graphics

CryEngine is the fastest high-end render in the world, with new features specifically designed for PC, Xbox 360 and Playstation3. Benchmark-setting graphical performance, near-photorealism in indoor and wide-open outdoor environments and extra-ordinary real-time special effects are some of the hallmarks of CryEngine technology. The CryEngine render provides seamless support for both indoor and outdoor environments on current platforms. Our multi-core and future-proof graphics technology ensure that CryEngine is next-gen ready. With CryEngine, scalability across multiple platforms is a further evolution to enable great looking games – regardless of the target platform.

2.6.1.2 Modular AI System

Realistically rendered and animated characters require state of the art AI systems to intelligently respond to the game environment and maintain the illusion of realism. CryEngine features powerful, scalable and flexible AI technology to handle character behaviors with modular sensory systems such as sight and hearing and fully support the complex requirements of the character locomotion system.

2.6.1.3 Realistic Characters

Characters and faces are some of the most important features of almost all modern AAA titles. CryEngine brings the most technically advanced (Figure 2.49), integrated and scalable animation and graphics technology together to deliver astonishingly real characters to cross-platform games at no extra licensing cost. As Crysis already demonstrated, Crytek are in the vanguard of the quest to achieve realistic characters in real-time graphics and we will of course continue this tradition with Crysis 2 and beyond.

Figure 2.49: Realistic Characters.

2.6.1.4 Live Create

CryEngine also introduces CryEngine Live Create™ (Simultaneous WYSIWYP on all platforms). This allows developers to work with a single editor, but see and play the results in real-time on PC, PlayStation and Xbox 360, hooked up to a single dev PC. The engine takes care of the conversion and optimization of assets in real-time, enabling instant, cross-platform changes to any part of game creation and, as a result, materially increasing the speed and quality, while significantly reducing the risk of multi-platform development.

2.6.2 User interface of CryEngine

2.6.2.1 First Launch

Game developers launch the Editor.exe located in either the Bin32 or Bin64 folder. They have to log in before enter to the interface (Figure 2.50) and CryEngine will be presented with an interface, as shown in the following Figure 2.51.

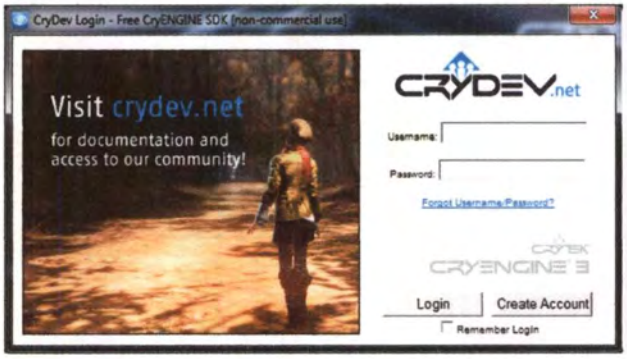

Figure 2.50: log in interface of CryEngine.

Figure 2.51: User interface of CryEngine.

2.6.2.2 Components of CryEngine user interface

The window where game developers can see the level is called the **Perspective Viewport window** (Figure 2.52). It is used as the main window to view and navigate your **level. This** is where a large majority of your level will be created and common tasks such as **object placement**, terrain editing, and in-editor play testing will be performed.

Figure 2.52: Perspective viewport window of CryEngine.

2.6.2.2.1 Control of CryEngine

The first step to interact with loaded level is to practice moving in the Perspective Viewport window.

1. Press **W** to move forwards.
2. Press **S** to move backwards.
3. **A** is pressed to move or strafe left.
4. **D** is pressed to move or strafe right.
5. Game developers can adjust the rotation of the camera by mouse (Figure 2.53).
6. When the viewport is the active window, hold down the right mouse button and move the mouse pointer to turn the view.
7. Game developers can also hold down the middle mouse button and move the mouse pointer to pan the view.
8. Roll the middle mouse button wheel to move the view forward or backward.
9. Finally, game developers can hold down Shift to double the speed of the viewport movements.

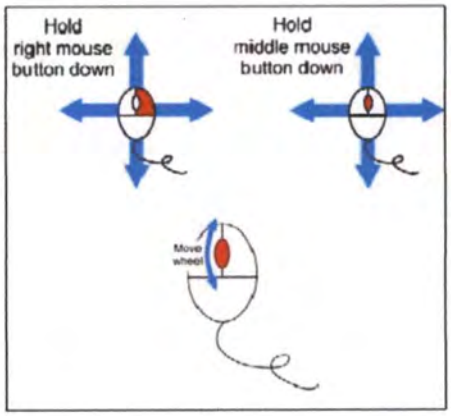

Figure 2.53: Using camera rotation by mouse.

Viewport movement speed control, The Speed input is used to increase or decrease the movement speed of all the movements. Game developers make in the main Perspective Viewport. The three buttons to the right of the Speed: inputs are quick links to the .1, 1, and 10 speeds (Figure 2.54).

Figure 2.54: Viewport movement speed control of CryEngine.

2.6.2.2.2 Toolbar

The toolbars, users can very quickly access many of the features of the sandbox editor by using simple icons and groups of icons at the top of the interface. These toolbars can be configured to fit the preferences and needs of individual users.

1. The **Standard Toolbar** contains open, save, hold, and fetch options (Figure 2.55).

Figure 2.55: Standard Toolbar of CryEngine.

2. The **Edit Mode Toolbar** contains various tools for level editing. These tools include undo and redo, link and unlink, select all, object movement/scaling, axes and terrain options, as well as object selection, saving, and loading (Figure 2.56).

Figure 2.56: Edit Mode Toolbar of CryEngine.

3. The **Terrain Toolbar** contains shortcuts to tools within the Terrain Editor, the Terrain Texture Layer editor, and Terrain Lighting dialog (Figure 2.57).

Figure 2.57: Terrain Toolbar of CryEngine.

2.6.2.2.3 Rollup Bar

Game developers will only use the **Objects** and **Entities** tab, which is the first and default tab within the **Rollup Bar** (Figure 2.58). To access the majority of scene elements, they will use this tab. It holds interfaces to the various Database libraries and the brush database on their hard drive. Game developers must have a pre-existing level opened in Sandbox to complete this recipe.

Figure 2.58: Rollup bar of CryEngine.

There are a variety of different sections contained in the Rollup Bar. The Rollup Bar that include:

1. The AI section

This section within the Rollup Bar contains AI control objects, which are used to control AI entities and their behaviors in the level. They can define a specific behavior for an AI with reference to its location or other variables in the level. AI control objects can define navigation paths or an area for the AI on the terrain, including boundaries and forbidden areas. The objects can be used by AI actors to perform specific actions or events, such as animations and changes of behavior.

2. The Area section

The Area section contains the area objects, which are used to create three-dimensional zones in the level that can be used to trigger events.

3. The Entities section

The Entities section contains all the entities, which the player can interact with in some form.

4. The Misc Objects section

The Misc Objects section includes various tools and functions used during a level's creation such as roads, rivers, and comments.

5. The Solids section

The Solids tool is used to create simple structures and objects, or placeholders for future art assets.

6. The Archetype entity section

The Archetype section allows users to access currently loaded archetype libraries within a given level. An Archetype entity is based on a regular entity and specifies individual parameter values for that regular entity. The main advantage of archetypes is that if the value of an archetype parameter is changed, all instances of that archetype in the level will be updated automatically. Archetype entities are organized into .xml libraries, which can be created in the editor under the Database view.

7. The Brush section

This will display a browser linked to the CryENGINE3/Game/Objects directory of game developers build. Brushes are compiled geometry containing no extra data other than collision. Typically, most levels are created with brushes as they are simple geometry.

8. The Geom entity section

The Geom entity section is a browser similar to the brush browser but instead allows for the placement of a very simple entity that takes its physicalization parameters from its assigned

geometry. When objects are placed as Geom entities and it have user-defined properties (discussed later), they become interactive entities with physical values, so they can behave like real-life objects. It is similar to a basic entity, but simpler, more efficient, and has fewer configurable parameters.

9. The Prefabs section

The Prefabs section contains the currently loaded Prefab libraries for a given level. Prefabs are groups of objects that can be placed in the level as instances similar to archetype entities. Altering one prefab universally applies the changes to each instance of the prefab object. Any iteration to the prefab is required to be saved to the Prefabs library to ensure they are correctly propagated across an entire series of levels.

10. The Sound section

This section contains a shortcut to the Entity/Sound section. This allows for the addition of specific sound entities to your level.

2.6.2.2.4 Console

The Console in the Sandbox editor is used to input variables. That shown the compile, error, and warning of game project (Figure 2.59)

Figure 2.59: Console of CryEngine.

2.6.2.2.5 Menus Bar

1. File

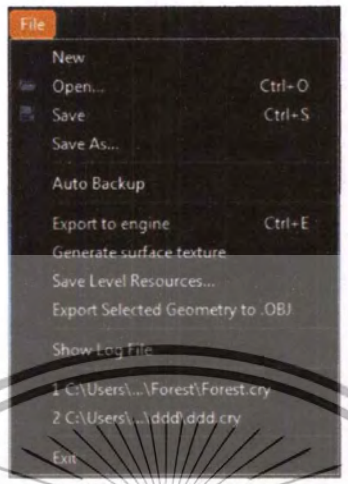

Figure 2.60:File menu of CryEngine.

2. Edit

Figure 2.61:Edit menu of CryEngine.

3. Modify

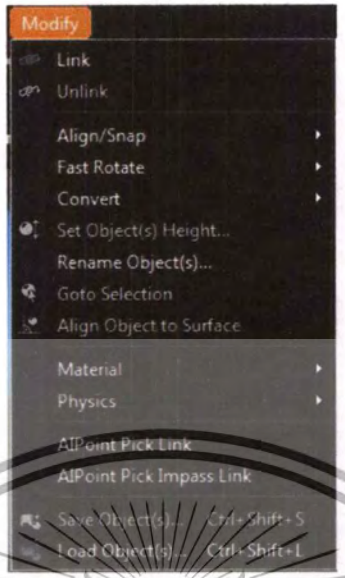

Figure 2.62: Modify menu of CryEngine.

4. Display

Figure 2.63: Display menu of CryEngine.

5. Config Spec

Figure 2.64: Config Spec menu of CryEngine.

6. Group

Figure 2.65: Group menu of CryEngine

7. Prefabs

Figure 2.66: Prefabs menu of CryEngine.

8. Terrain

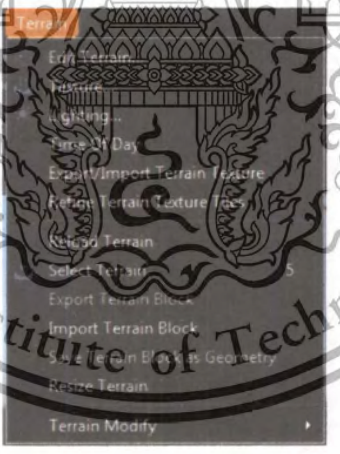

Figure 2.67: Terrain menu of CryEngine.

9. Sound

Figure 2.68:Sound menu of CryEngine.

10. Game

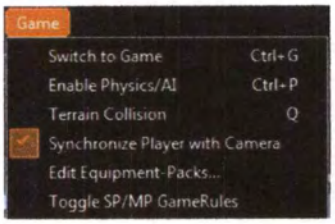

Figure 2.69: Game menu of CryEngine

11. AI

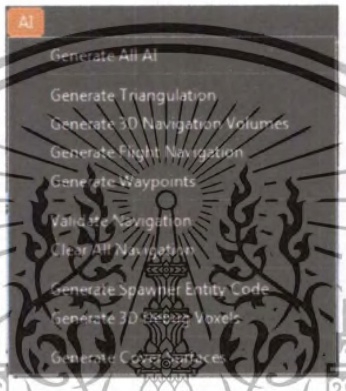

Figure 2.70: AI menu of CryEngine.

12. Clouds

Figure 2.71: Clouds menu of CryEngine.

11. Tools

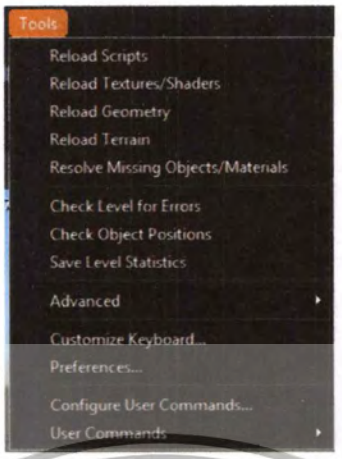

Figure 2.72: Tools menu of CryEngine.

12. View

Figure 2.73: View menu of CryEngine.

13. Help

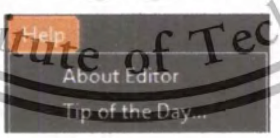

Figure 2.74: Help menu of CryEngine.

Chapter 3

Evaluation of Game Engines

3.1 Criterion of Game Engines Evaluation

All game engines were tested for game developers who are the beginner to create game project by examination of five criterions as the following,

3.1.1 Editing

Is well the user interface? Is the editor easy to use? How can manage custom the user interface? What about step to create game? How is the editor Started?

3.1.2 Graphic

Which are the 3D programs that each game engine supports? What are object graphic file that each game engine supported and can add assets in to engine?

3.1.3 Platform

Which are publishing platform that game engines supported? Are there have current platform for game developers to use?

3.1.4 Content

How easy is the process of including game content as well as external models into the map? Which restrictions have to be considered when importing custom models?

3.1.5 Game Play

What happen when the game is run about moving in game, component in game and view in game?

3.2 Game Project Design

The Game Project was designed with three types of Game Engines which have the similar map for comparison based on the five criterions.

3.2.1 GameCore Map Design

We designed with terrain that has mountains, trees, player, cabin, water, dragon as shown in Figure 3.1:

Figure 3.1: GameCore map design.

3.2.2 Unity Map Design

We designed with terrain that has mountains, trees, player, cabin, water, dragon as shown in Figure 3.2:

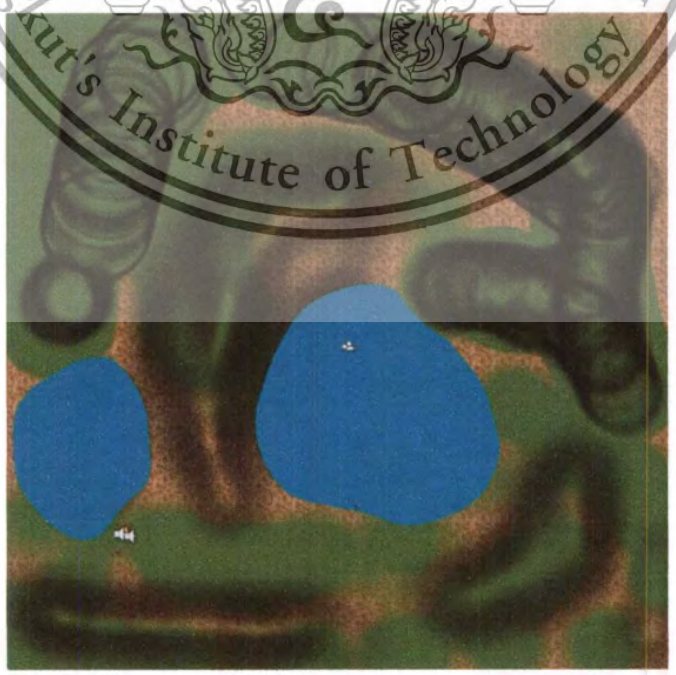

Figure 3.2: Unity map design

3.2.3 CryEngine Map Design

We designed with terrain that has mountains, trees, player, cabin, water, dragon as shown in Figure 3.3:

Figure 3.3: CryEngine map design

3.3 Comparison of Game Engines Evaluation

The comparison of three game engines based on five criterions are demonstrated as the following,

3.3.1 Comparison of Editing

The creating and editing steps of terrain used in these three game engines are demonstrated and compared as below.

3.3.1.1 GameCore Editing

GameCore is not stared as a standalone program. It does not incorporate texture browser, script editor, model viewer and other component necessary for editing. Game developers have to add all objects, asset, texture and other component. Game developers have to find or create them by you.

User interface of GameCore is similar with other game engines (see figure 3.4) but cannot customize the layout likes Unity 3D. GameCore has user interface that easy to use but some functions have several step to access them and it does not much difficult to use.

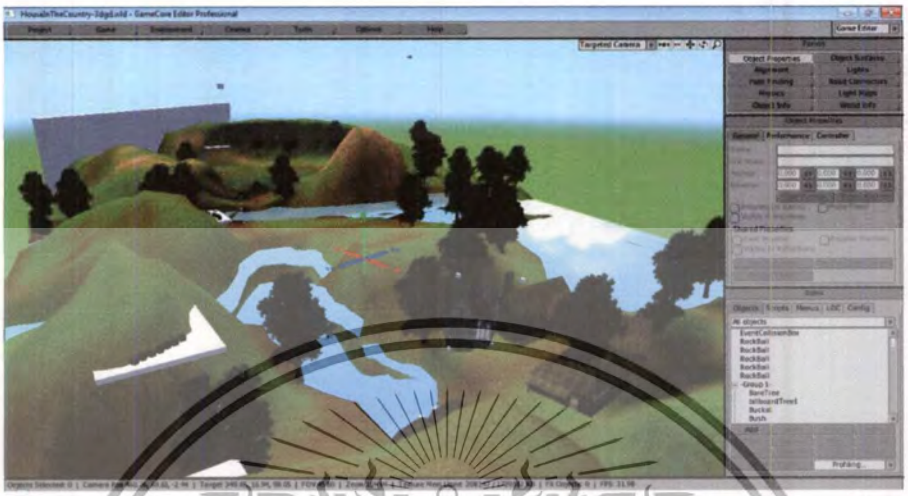

Figure 3.4: GameCore user interface.

The step for create project game of GameCore we will show example of creating terrain and edit terrain for one level in game to compare with Unity and CryEngine. Such as when game developers need to create terrain of their game, they can click on **"Add -> From Template -> [Terrain] -> Open"** after that terrain will appear on the screen of GameCore (see figure 3.5 and 3.6). They used 4 step for create terrain in game. They can edit terrain by click on **"Edit Terrain"** on **Panels** layout and then they can rise and lower, set high, smooth, noise, and etc. at **Bush Mode** on **Painting of Terrain**. They can specify brush size, softness, and brush strength (see figure 3.7 and 3.8).

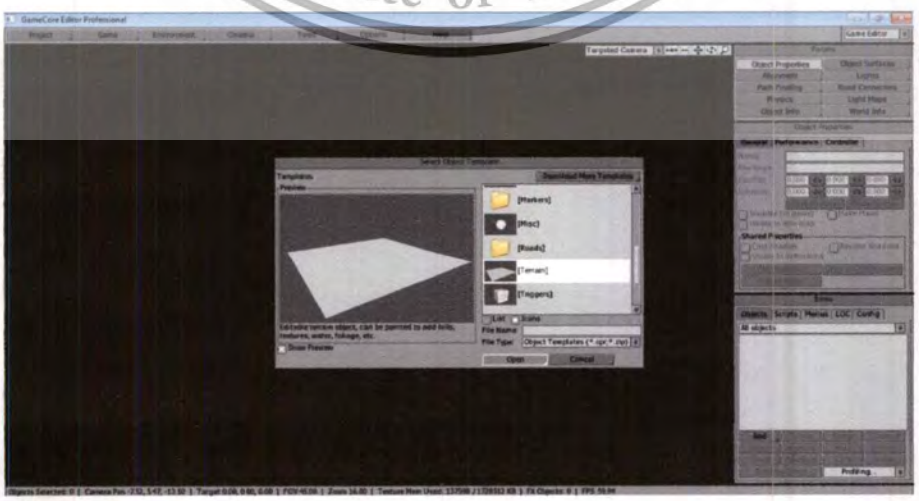

Figure 3.5: Adding terrain of GameCore.

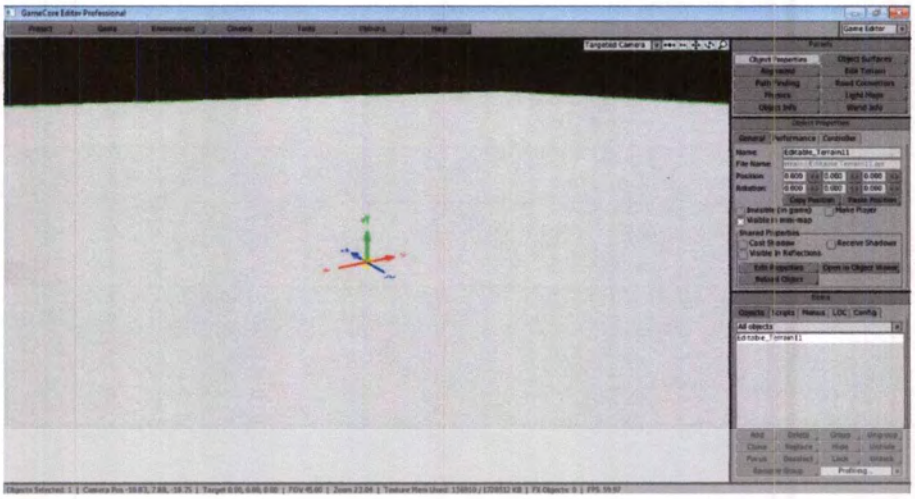

Figure 3.6: Terrain of GameCore.

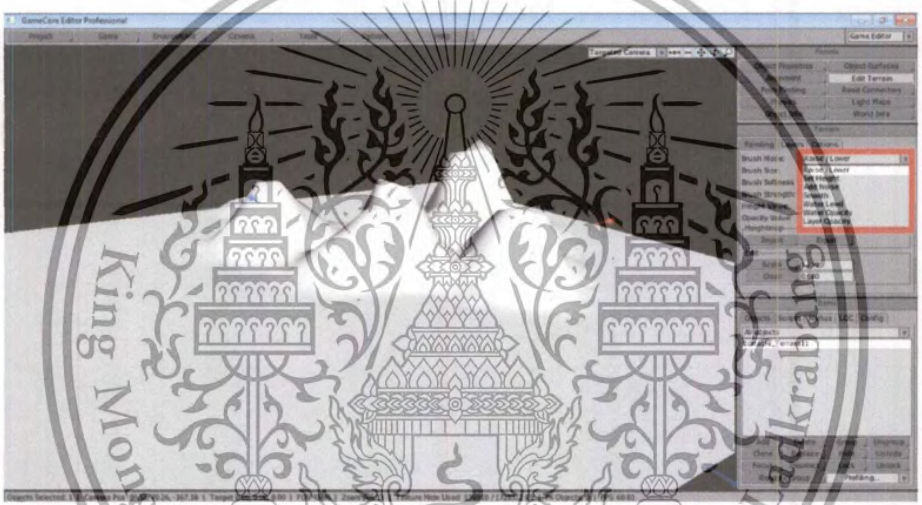

Figure 3.7: Editing terrain and brush mode of GameCore.

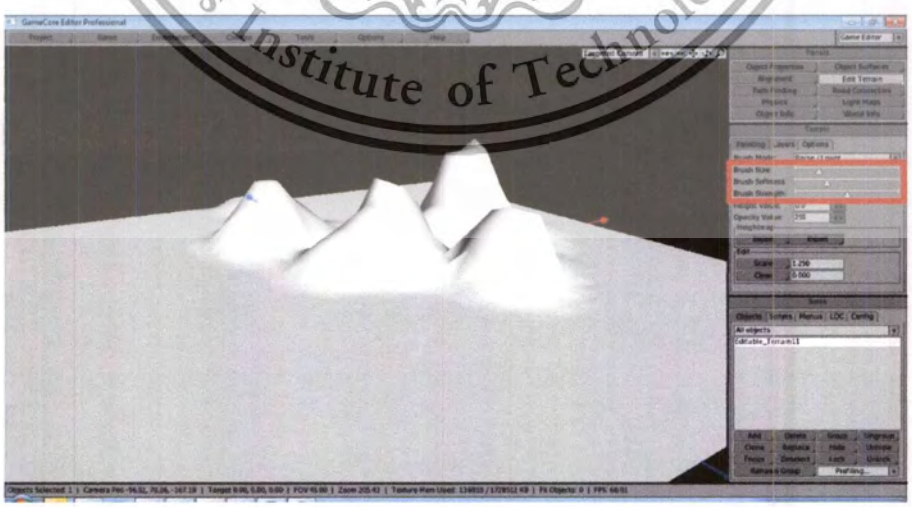

Figure 3.8: Brush size, softness, and brush strength of GameCore.

Game developers can add texture by go to **Layers** on **Terrain** layout to add layer for edit surface to select the texture in **Diffuse Map** (see figure 3.9 and 3.10). After that they change brush mode to **Layer Opacity** and then they can paint texture on the terrain (see figure 3.11 and 3.12). If they need several textures they have to add several layers for each texture that they need to paint into terrain (One layer per one texture). Then game developers have to used more 5 steps to edit terrain and each steps depended on game developers who wanted to edit their terrain on game.

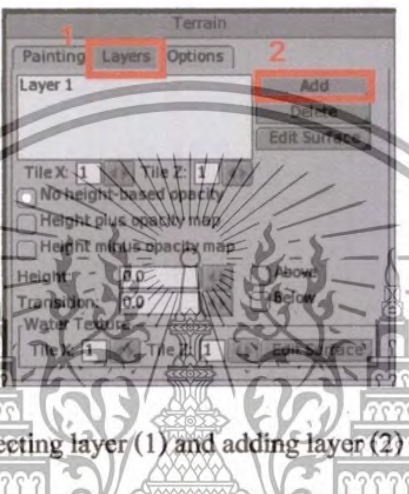

Figure 3.9: Selecting layer (1) and adding layer (2) of GameCore.

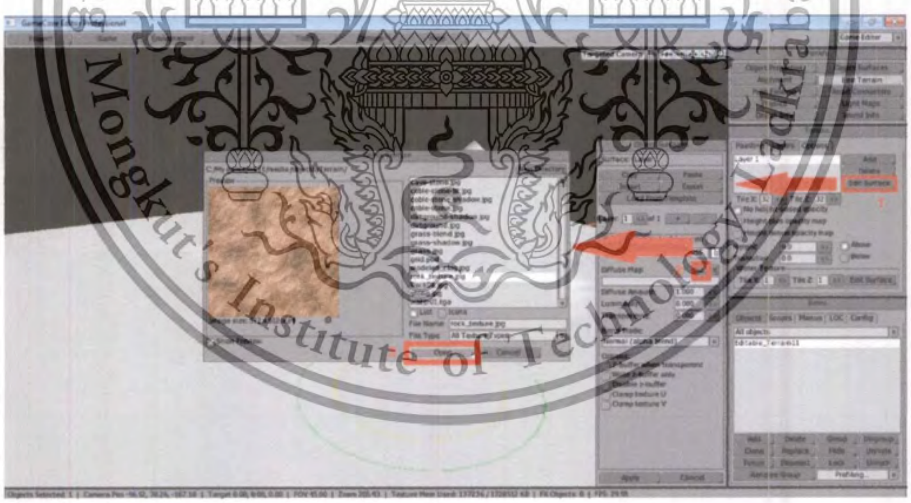

Figure 3.10: Step adding texture of GameCore.

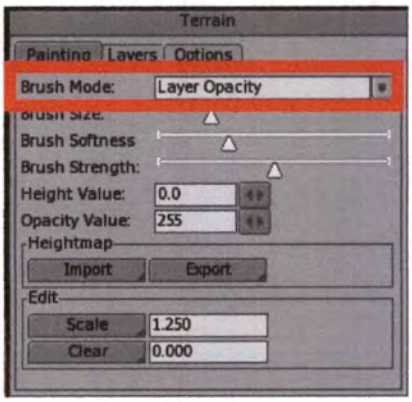

Figure 3.11: Changing Brush Mode to Layer Opacity of GameCore.

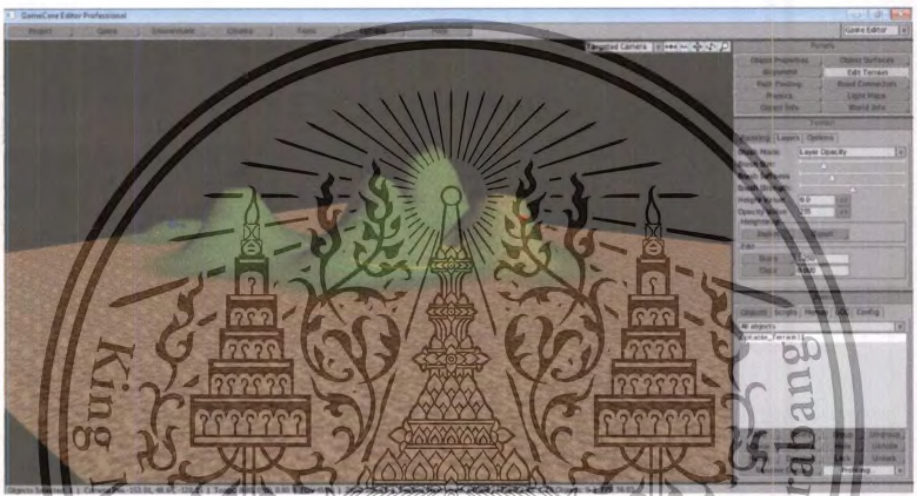

Figure 3.12: Painting texture on the terrain of GameCore.

3.3.1.2 Unity Editing

Unity is started as a PC and MAC standalone program. It incorporated texture browser, script editor, model viewer and other component necessary for editing.

The user interface of Unity is good design that similar with other game engines and other game engines (Figure 3.13). When game developers used each layout, they can understand the interface easily and easy to use each function of Unity. Game developers can customize the layout to another area that they want . For example, game developers can move project view and hierarchy to the right side of Unity (Figure 3.14) or floating editor window (Figure 3.15) and the can change more that they need to manage.

Figure 3.13: Unity user interface.

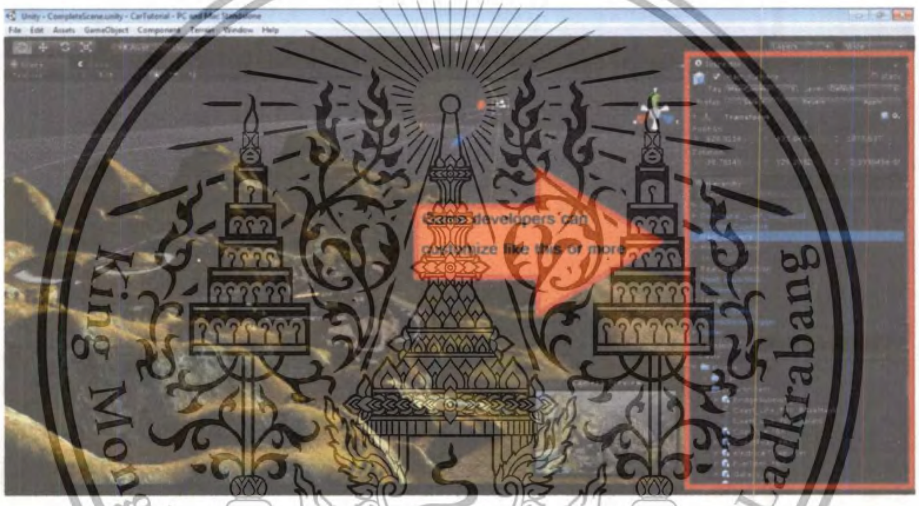

Figure 3.14: Unity custom layout management.

Figure 3.15: Floating editor window of Unity editor.

The step for create project game of Unity we will show example of creating terrain and edit terrain for one level in game to compare with GameCore and CryEngine. Such as when game developers need to create terrain, they click on **Terrain** on the tap menu bar and click on **Create Terrain** (see Figure 3.16). Then the terrain will appear on the scene (see Figure 3.17) and game developers used 2 steps for created terrain. If game developers need to edit the terrain, they can use tools in the inspector to raise and lower, set the terrain height, and smooth the terrain height which have many brushes and can specify brush size and opacity (see Figure 3.18). Game developers can add texture into terrain by used **Pain the terrain texture** tool to paint on the terrain and they have to add textures on **Edit Texture** before (see Figure 3.19). They can add many textures to use. After that game developers can paint textures on the terrain (see figure 3.20). Then game developers have to use more 3 steps to edit terrain and each steps depended on game developers who wanted to edit their terrain on game.

Figure 3.16: Creating terrain of Unity editor.

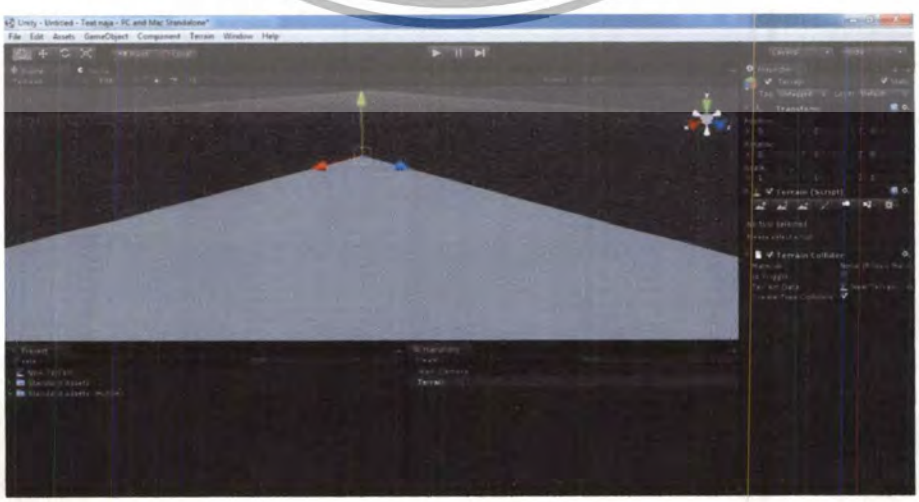

Figure 3.17: Terrain of Unity editor.

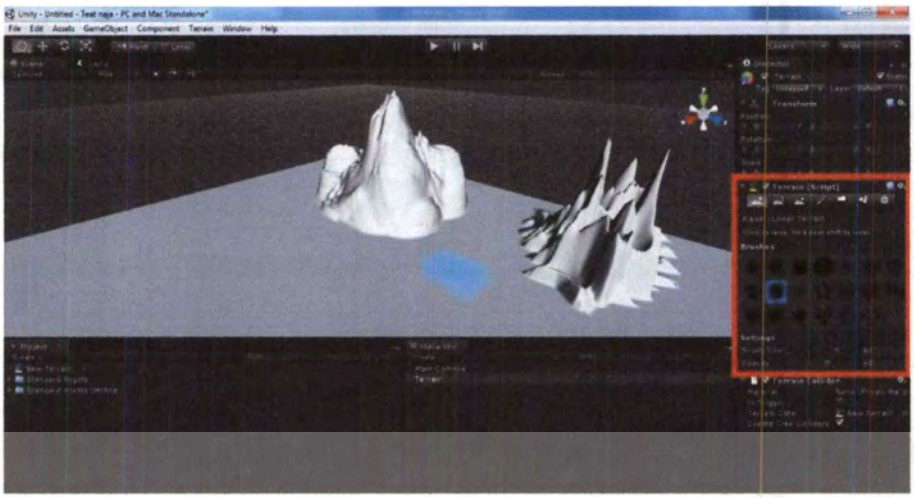

Figure 3.18: Editing the terrain of Unity editor.

Figure 3.19: Adding texture of unity editor.

Figure 3.20: Painting the terrain of Unity editor.

3.3.1.3 CryEngine Editing

The user interface of CryEngine is show in Figure 3.21. CryEngine has the toolbar and menus that enough and easy to use to create game. The complication of the game creator can be dissolved by using the right layouts, more convenient and easier.

To create terrain, click on **File -> New** (Figure 3.22). Game developers will get window in Figure 3.23 that they can set size of terrain and name of project. Thus, game developers have to use 3 steps for creating terrain. Game developers can edit terrain by click on the icon in Figure 3.24 then use function terrain of rollup bar to edit terrain (Figure 3.25). Game developers have used more 2 steps for editing terrain that depends on they wanted to edit terrain. In function terrain of rollup bar has modify, vegetation, layer painter, voxel painter, holes, environment, move area and mini map function. All of these function use to edit terrain. Example, click on modify and choose brush setting to flatten, set size of brush and high of terrain when brush paint on it (Figure 3.26 and 3.27). Click on texture icon to add texture on terrain (see Figure 3.28). Game developers will get the window in Figure 3.29. They can choose the texture that they like to create the texture of your terrain (see Figure 3.30). Then click on change layer texture to choose texture for painting terrain (see Figure 3.31 and 3.33) and use the right layouts to paint terrain (see Figure 3.33).

Figure 3.21: User Interface of CryEngine

Figure 3.22: Click on File > New to create project .

Figure 3.23: 1) Set name of project 2) Set size of terrain 3) Multiple of size of terrain.

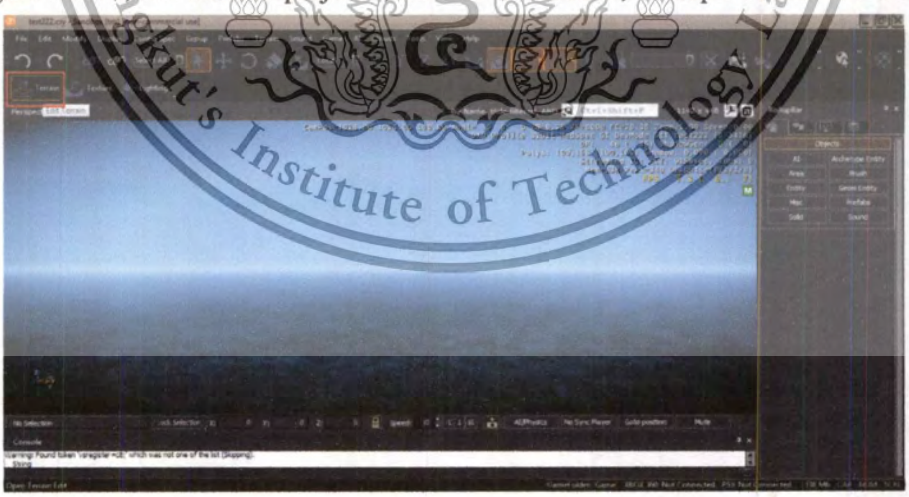

Figure 3.24: Click on terrain icon to edit terrain.

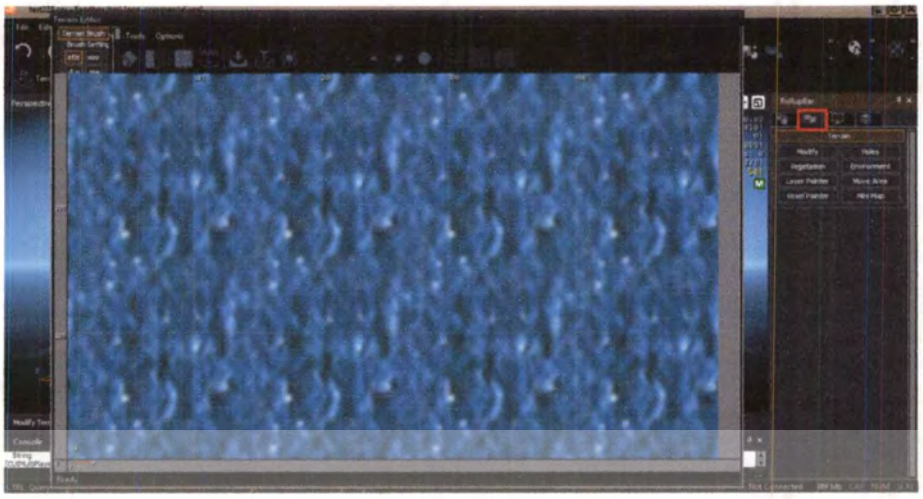

Figure 3.25: Use function terrain of rollup bar to edit terrain.

Figure 3.26: 1) select modify 2) Choose brush setting to flatten 3) set size of brush and high of terrain when brush paint on it.

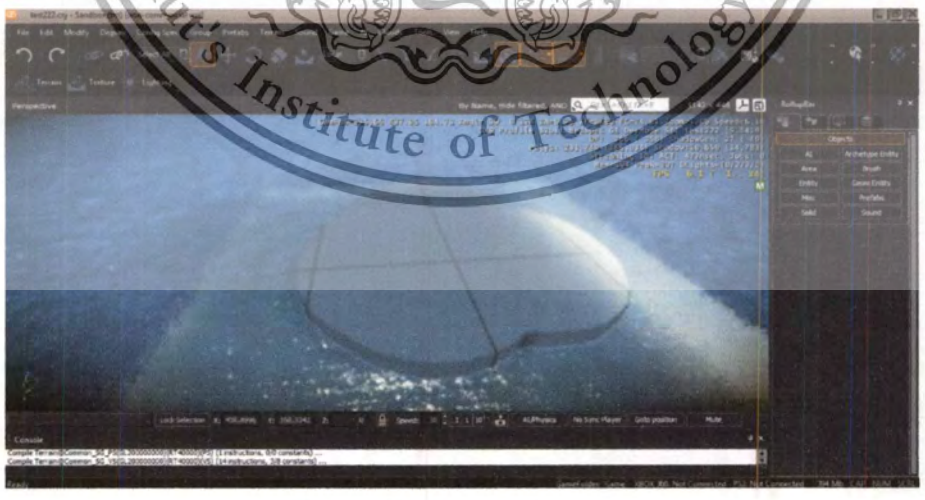

Figure 3.27: Result of terrain.

Figure 3.28: Click on texture icon to add texture to terrain.

Figure 3.29: Terrain texture layers, Click on materials/material terrain.

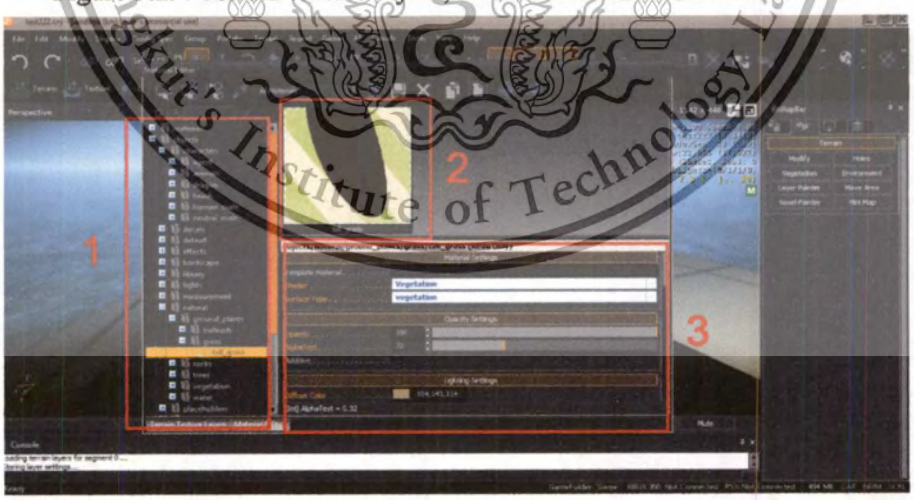

Figure 3.30: 1) choose background of terrain 2) sample of background 3) detail of background.

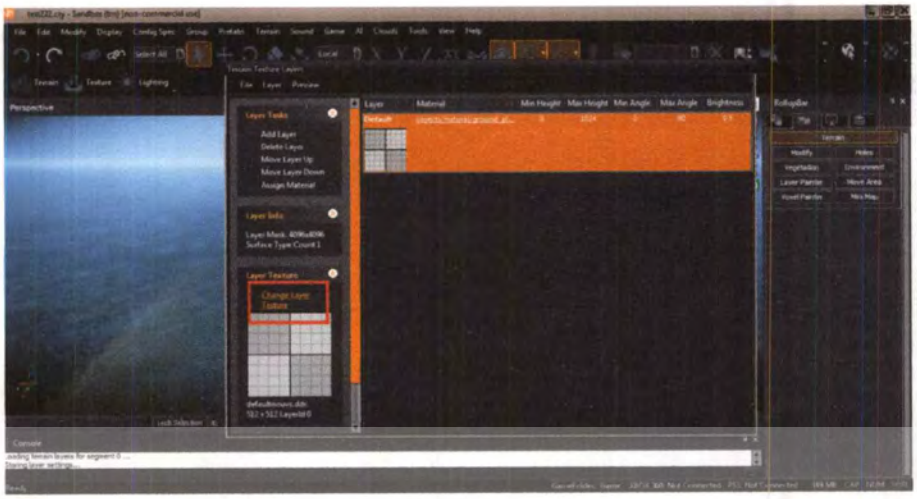

Figure 3.31: Click on change layer texture.

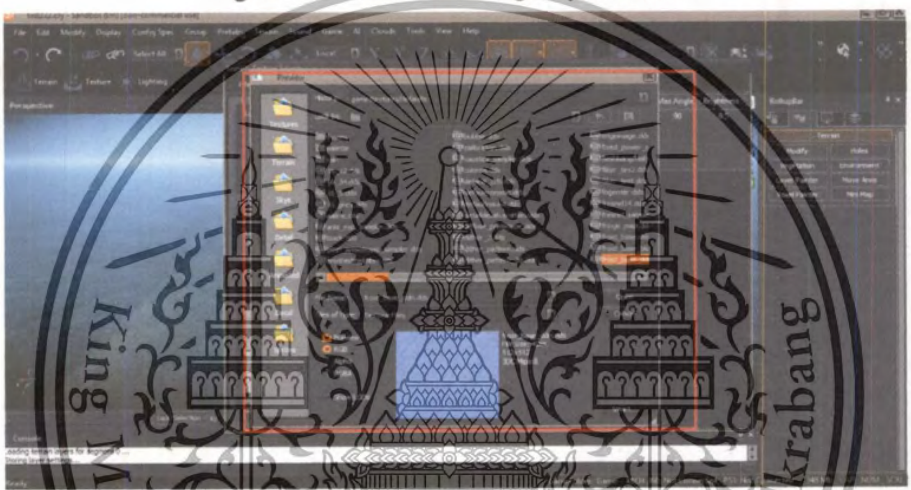

Figure 3.32: Choose texture for painting terrain.

Figure 3.33: 1) select layer painter 2) set size and attribute of brush 3) paint terrain.

3.3.2 Comparison of Graphic

To import either 3D model files or 3D animation programs to support game engines are describe as below.

3.3.2.1 GameCore Graphic

GameCore supports every 3D modeling package available through other native support for the most common, industry standard formats like Alias OBJ, 3D Studio (.3ds) and Autodesk FBX, including:

- Autodesk 3DS Max (.fbx, .3ds)
- Autodesk Maya (.fbx, .obj)
- Autodesk Motion Builder (.fbx)
- Lightwave (.lwo, .lws)
- Milkshape 3D (.ms3d, .obj, .lwo, .3ds)
- SoftImage (.obj, .fbx)
- Silo 3D (.3ds, .obj)
- Google SketchUp Pro (.3ds, .obj, .fbx)

3.3.2.2 Unity Graphic

Unity support many 3D modals and major tools are supported that can be used with Unity such as:

- Maya
- Cinema 4D
- 3DS Max
- Cheetah 3D
- Modo
- Lightwave
- Blender

All assets in Unity project can read file:

- .FBX
- .dae
- .3ds
- .dxf
- .obj

3.3.2.3 CryEngine Graphic

CryEngine support many 3D models such as:

- Autodesk 3DS Max
- Autodesk Maya
- Autodesk Motion Builder
- Milkshape 3D
- SoftImage
- Silo 3D
- Google SketchUp Pro

All assets in CryEngine can read file:

- .max
- .3ds
- .obj
- .fbx
- .ma

3.3.3 Comparison of Platform

The aim of platform comparison is to investigate whether the several platforms can be installed into game engine including several computers can open this game.

3.3.3.1 GameCore Platform

GameCore currently supports publishing to the following platforms:

- Windows (XP, Vista, Version 7)
- Mac OS (Version 0.4 or higher)
- Web 3D publishing (all GameCore Version)

3.3.3.2 Unity Platform

The Asset Server is available as Mac OS X and Microsoft Windows Installers and as Linux RPM, Debian, and source code packages.

3.3.3.3 CryEngine Platform

CryEngine supports to these following platforms:

- Windows
- Mac OS
- Web 3D publishing
- Linux
- Xbox
- Play Station

3.3.4 Comparison of Content

The purpose of content comparison is to verify the easiness of editors. After the external model were imported into game project of each editors, there are restrictions of importing custom models as below explanation.

3.3.4.1 GameCore Content

When game developers want to add game object, they can click on **ADD -> From object file** (Figure 3.34) and then they can select file from folder (Figure 3.35) and the object will appear in the map. When they have added game object already they cannot specify position in the map of the game object but it will appear in middle of GameCore editor scene. When they run the game, player cannot collide with the object that they add but the player walk through the object.

GameCore editor does not have effect. If game developers want some effect, they will have to import effect file like adding game object that mean they have to create animation file for the effect in game. For example, the dragon can flame that game developers have to make the animation file for flame.

GameCore editor will saved the map with .wld (world file) that is the surname file of GameCore editor. The game file that game developer saved that can open with GameCore editor of other computers but they have to save all file that related with their game such as game object, audio file, world file, and etc. If they added game object from other folder and they did not save that objects file, it would not see that objects and would be error.

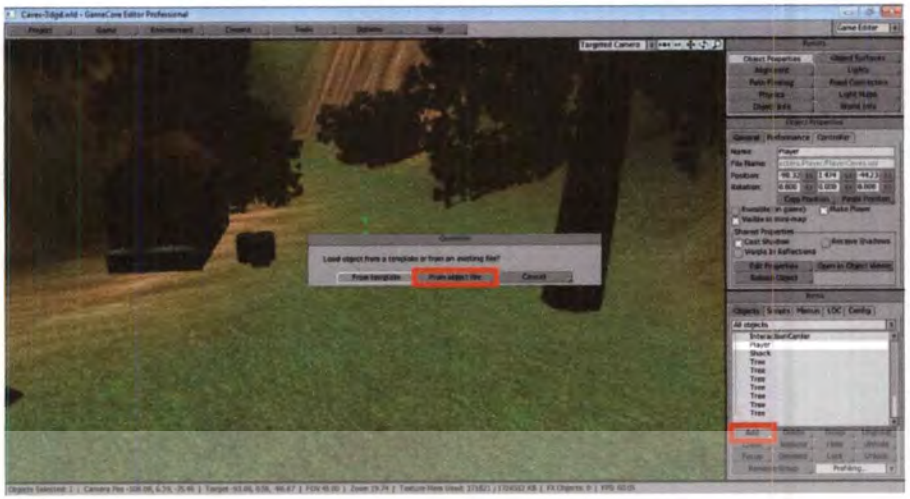

Figure 3.34: Adding game object of GameCore editor.

Figure 3.35: Selecting object from the folder of GameCore editor.

3.3.4.2 Unity Content

When game developers want to import game object from other folder, they had to click **Asset** on menu bar and then click on **Import New Asset** (Figure 3.36) after that they can select the object file from their folder (Figure 3.37). The game object will appear in the project panel and game developers can specify the position of the game object by drag the game object from project panel to the scene. When game developers run the game, player cannot collide with the game object that they imported but the player walk pierce through the object. But they can made collision by click on the **Generate Collider** in the inspector of game object file (Figure 3.38).

Unity editor have some special effect that is **"Particle System"** like the smoke, game developers can apply for their game (figure 3.39). For example game developers can use particle system for fire of volcano, bonfire, flame of dragon in this special project, and etc. Game developers can edit size and color of particle system in the inspector and they can import texture

for reality. In the unity editor that have many virtual natures like waterfall, fountain, wave, storm, snow, bubble, bonfire, and etc. That was in the **"Particles Folder"** of **"Standard Asset Folder"**.

Map file of unity will in the folder that game developers created in their computer that include all game file that related with their game. When game developers open the map file, they can open with their folder and unity will open all file of their game such as game object, scene, animation, script, etc. Game developers can copy that folder for open with other unity editors of other computers and all of the game will appear in other computers. Otherwise game developers import asset from the different folder, they can open and run that game.

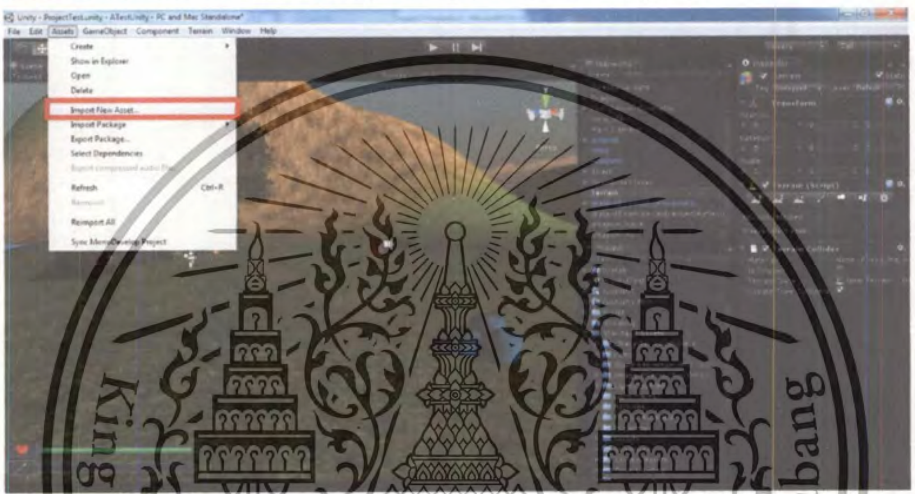

Figure 3.36: Importing the asset of unity editor.

Figure 3.37: Selecting game object of unity editor.

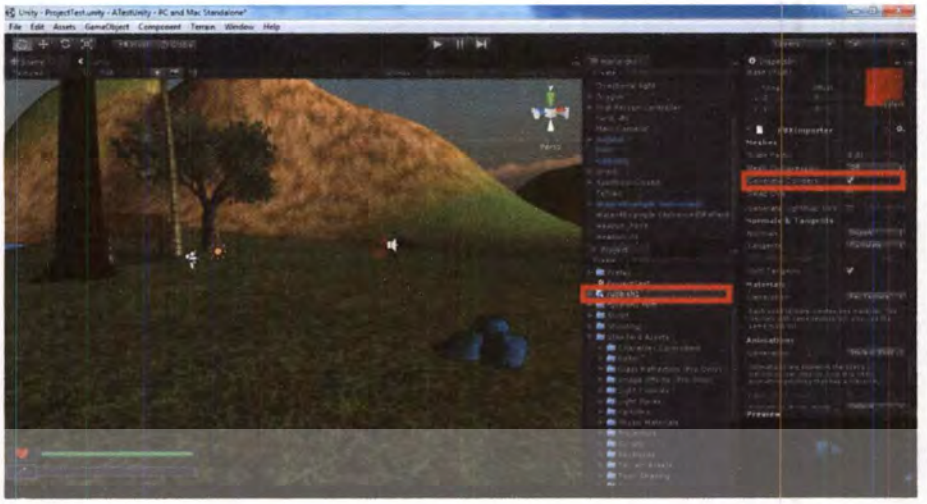

Figure 3.38: Generating collider of unity editor.

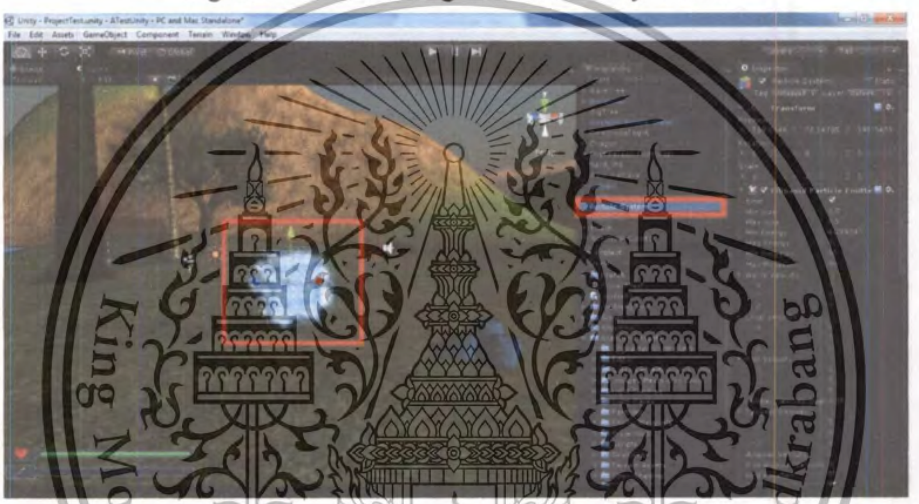

Figure 3.39: Particle System.

3.3.4.3 CryEngine Content

When game developers want to add object into game project, they can click on function objects from rollup bar and choose entity. The function browser will appear on CryEngine Screen. The object classify to each folder. In Figure 3.40 select the AI folder and choose Flyer. The object represent mouse. Game developers can move object to anywhere that they want. Then click to paste the object (Figure 3.41). If game developers want to add external game object from 3ds max or Maya to map of game project, they have to install plug-in with 3ds max or Maya for exported the model to CryEngine (Figure 3.42) and save 3d model file into the folder of CryEngine. Instantly, the game developers run the game project they can be collided to the game object because CryEngine editor can make collision by itself.

CryEngine editor have many special effect that is **"Particle Effect"** like smoke, fire, stream, and etc. That was in **"DtaBase View"** of CryEngine editor (Figure 3.43). Besides the particle effect, CryEngine editor have many virtual environments such as rain, shack, and tornado

that were in **"Environment"** folder of **"Entity"** in roll up bar (Figure 3.44). Game developers can use the special effect and virtual environments in their game projects (Figure 3.45).

Figure 3.40: 1) Select objects function 2) choose Entity 3) function browser will appear 4) Click folder AI and choose Flyer.

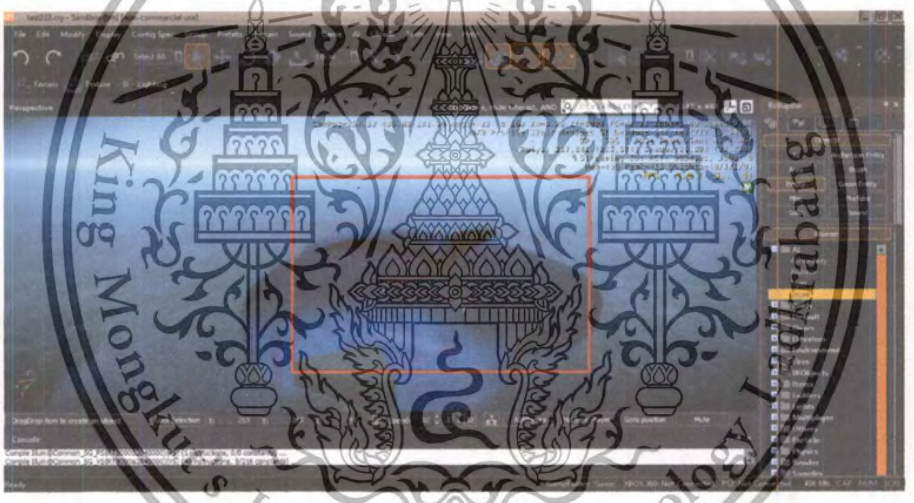

Figure 3.41: Adding object to map of CryEngine.

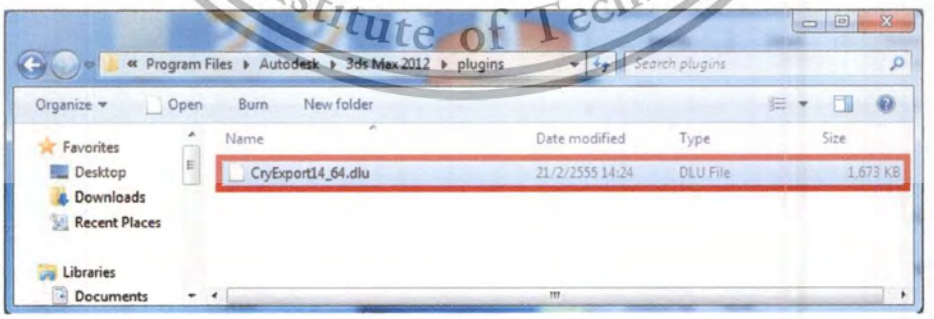

Figure 3.42: Plugins of 3ds max.

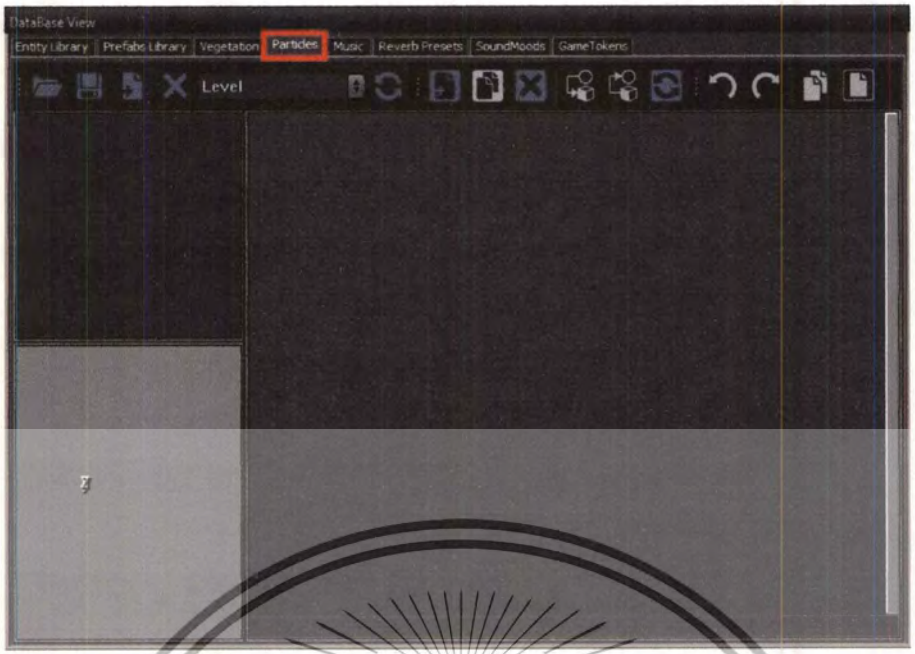

Figure 3.43: "DataBase view" of CryEngine editor.

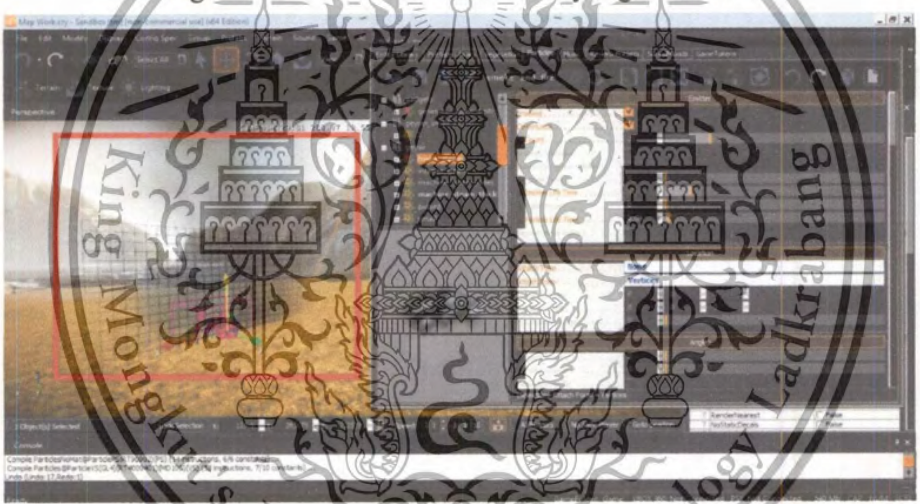

Figure 3.44: Particle effect in "DataBase View" of CryEngine editor.

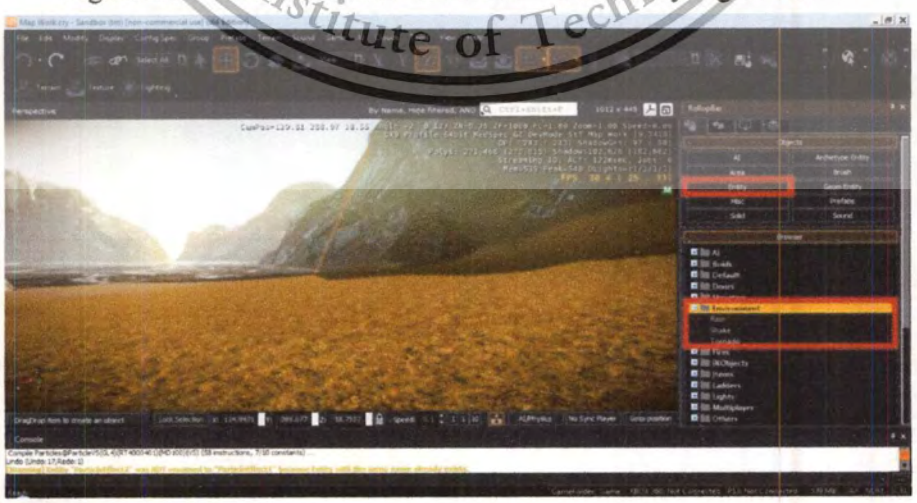

Figure 3.45: Virtual environments of CryEngine editor.

CryEngine will save the map with .cry and create another folder that has the same name (Figure 3.46) to keep the object and effect in map. When game developers want to open this map with another computer, they must copy both of file in folder and other objects that they imported from external CryEngine editor. If they do not copy all files, map will not be opened.

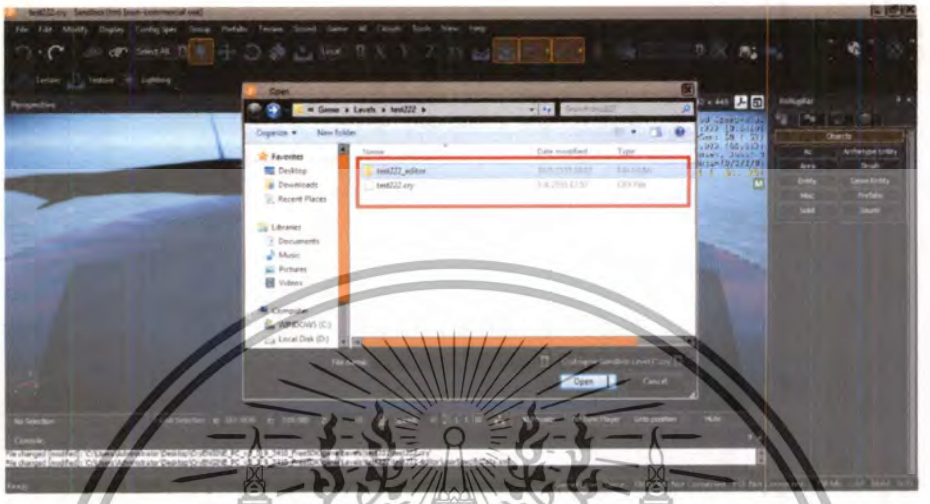

Figure 3.46: Project file in CryEngine folder.

3.3.5 Comparison of Game Play

Game play comparison of each game engine are performed during we run the game project. We can check game movement of player, component in game, and also check the view in game in order to compare these three game engines.

3.3.5.1 GameCore Game Play

If game developers want to run game in GameCore editor, they can go to **Game** on the menu bar and then click on the **Run Game** button after that they can select the map that they want to run (Figure 3.47). Game developers run the game which appear the scene from camera of player that they specify position in map.

Movement in game of player is normal and it depends on the speed of player walking that can edit in script of player. For example, **"ControllSetting Walkspeed"** that is the script for control walk speed of player (Figure 3.48).

If the GameCore editor appears error that will appear when game developers run the game. GameCore editor will show **"Script error"** (Figure 3.49) when game developers have something wrong but it did not show what the script error and they cannot know how to edit.

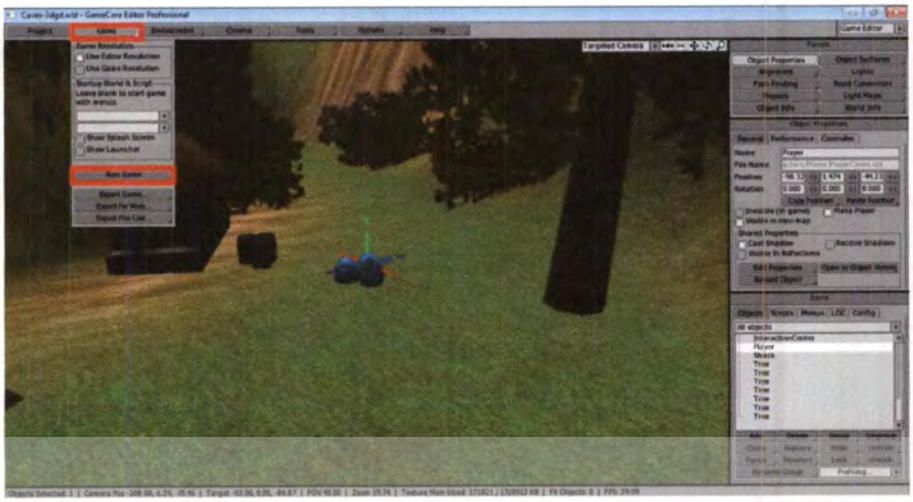

Figure 3.47: Running game of GameCore editor.

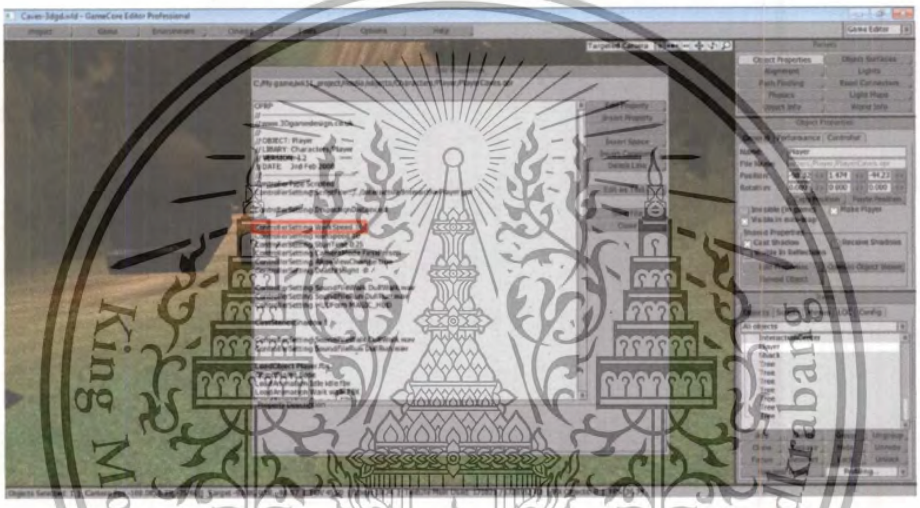

Figure 3.48: Controlling walk speed of GameCore editor.

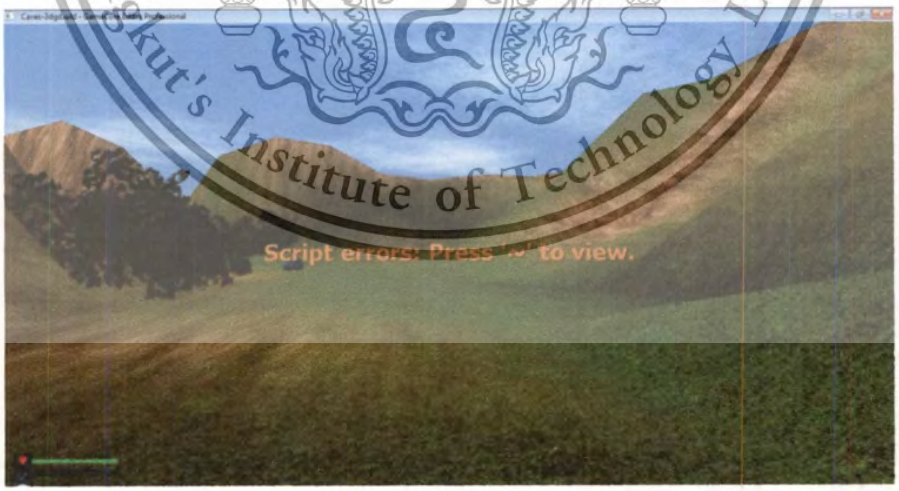

Figure 3.49: Script error message.

3.3.5.2 Unity Game Play

When game developers want to run the game, they can click on the **PLAY** button (Figure 3.50) and they want to stop the running game they can click as before. At the time that Game

developers run the game that will appear the scene that from camera of player that they specify position in map.

Movement in game of player is normal and it depend on the speed of player walking that can edit in the inspector of first person controller in the hierarchy panel (Figure 3.51).

When the unity editor appears error, it will appear in the lower bar of unity editor (figure 3.50). It shows the error of script and when the error happen, game developers cannot run the game and they have to edit the script before running again (Figure 3.52).

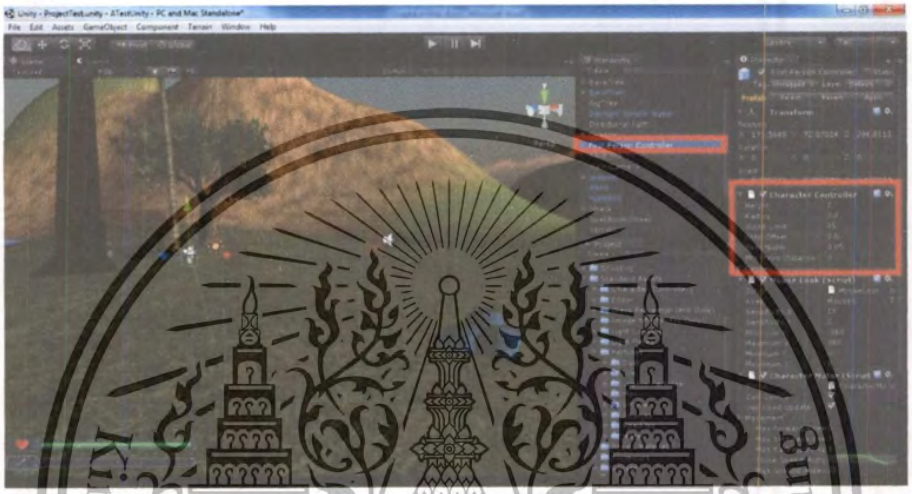

Figure 3.50: Editing walk speed of Unity editor.

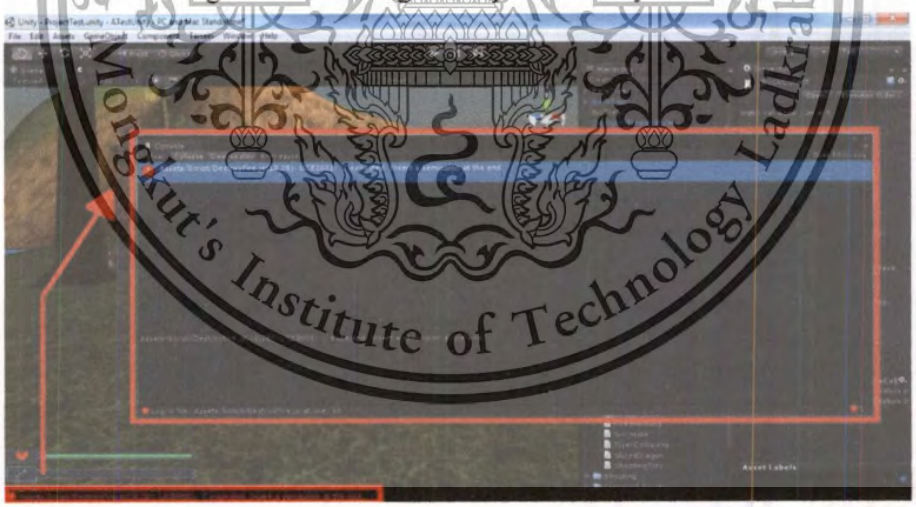

Figure 3.51: Script error of Unity editor.

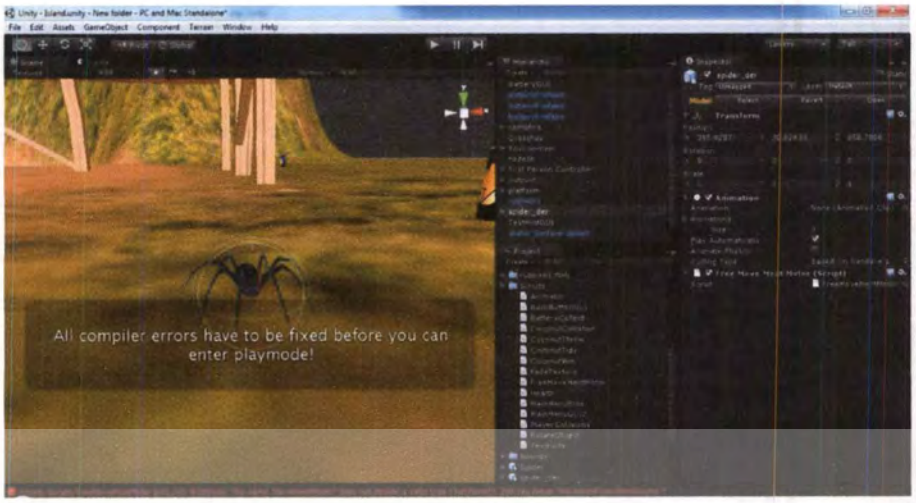

Figure 3.52: Compiler error of Unity editor.

3.3.5.3 CryEngine Game Play

When game developers want to run game project in CryEngine editor, they can click on **Game** at top menu bar and select **switch to game** or push **Ctrl + G** (Figure 3.53). After game developers run the game project, CryEngine running will appear the scene that from position of play that game developers specify **"SpawnPoint"** in multiplayer of entity in roll up bar (Figure 3.54).Movement in game of player is normal and it depend on the speed of player walking that CryEngine editor had specified the speed of player already. Cryengine is a real time program. If game developers make some mistake, Cryengine will show error at console (Figure 3.55).

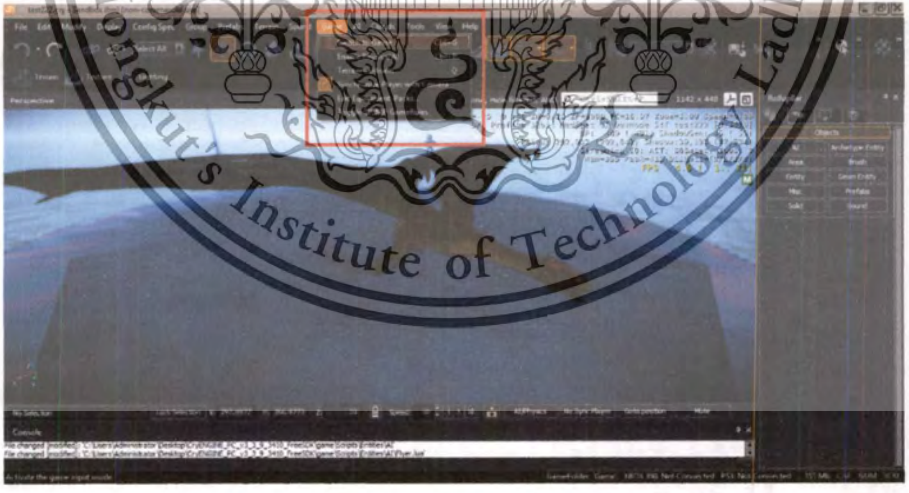

Figure 3.53: Running game project of CryEngine.

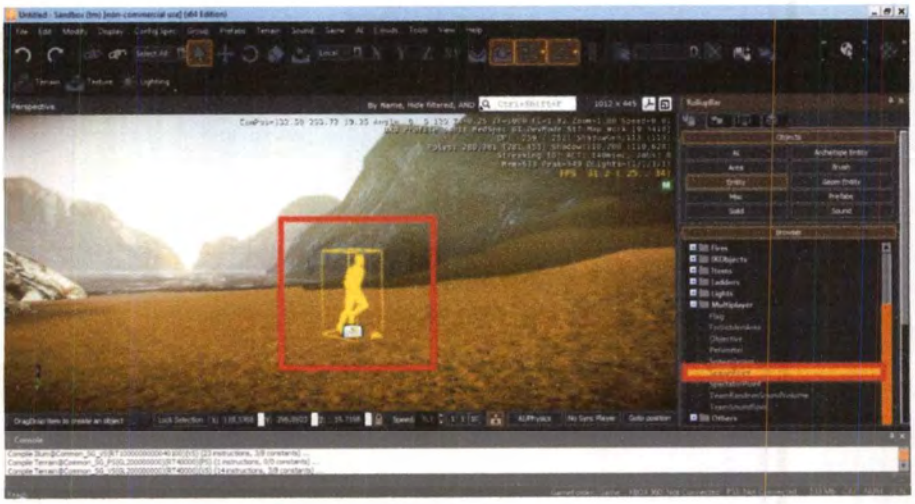

Figure 3.54: Specified player by "SpawnPoint"

Figure 3.55: Errors at console of CryEngine.

Chapter 4

Result of Evaluation of Game Engines

This chapter shows the result of evaluation of game engines that we compare three game engines with five criterions as editing, graphic, platform, content, and game play.

4.1 Result of Editing Criterion

In editing criterion we compare the user interface design, creating terrain, editing terrain. First we compare user interface design with three engines as GameCore, Unity CryEngine. User interface of GameCore has good design that has many toolbars and menus. Game developers cannot customize the layout of editor and some menus have many steps to access the function such as when we want to run game project, we have to crick on **Game** and then click **Run Game** and select the game level that we want to run which is more complexity. User interface of Unity is better design than GameCore and game developers can customize the layout of editor. There are many toolbars and menus to access the function and toolbar has icon to access direct function. User interface of CryEngine is a good design as Unity and better than GameCore. Game developers can customize the layout of editor. CryEngine has many menus and toolbars that enough and easy to create game project.

Second we compare creating terrain of each game engine that are the first time of creating our game project and each game engines has difference way to create terrain. For GameCore we have to create terrain 4 steps and we show step to create terrain of GomeCore in section 3.3.1.1 of chapter 3. Unity has 2 steps to create terrain and we show the step to create terrain of Unity in section 3.3.1.2 of chapter 3. The last game engine, CryEngine has 3 steps to create terrain and we show the step to create terrain of CryEngine in section 3.3.1.3 of chapter 3.

The last part of editing criterion is editing terrain that second time to create our game project and each game engine has difference way to edit terrain. For GameCore we have to edit terrain 5 more steps that depend on game developers want to do the game project and we show step to edit terrain of GomeCore in section 3.3.1.1 of chapter 3. Unity has 3 more steps to edit terrain that depend on game developers want to do the game project and we show the step to edit terrain of Unity in section 3.3.1.2 of chapter 3. The last game engine, CryEngine has 2 more steps

to edit terrain that depend on game developers who want to do the game project and we show the step to edit terrain of CryEngine in section 3.3.1.3 of chapter 3.

We conclude the result of edit criterion of all parts and all engines that we mentioned above in Table 4.1.

Table 4.1: The Result of Editing Criterion

Editing	GameCore	Unity	CryEngine
User Interface design	- Good design. - cannot customize layout.	- Best design. - can customize layout.	- Best design. - can customize layout.
Creating Terrain	- use 4 steps.	- use 2 steps.	- use 3 steps.
Editing Terrain	- have 5 steps or more.	- have 3 steps or more.	-have 2 steps or more.

4.2 Result of Graphic Criterion

In graphic criterion we compare only the 3D programs and 3D files that supported for each game engine which shown in Table 4.2. The 3D programs that each game engine supported of each game engine have many programs and CryEngine support all 3D programs while GameCore and Unity support some 3D programs. The 3D files that each game engine can import the 3D model into the game project which CryEngine support all 3D files while GameCore and Unity support some 3D files.

Table 4.2: The Result of Graphic Criterion

Program Supported	GameCore	Unity3D	CryEngine
- Autodesk 3ds Max	√	√	√
- Autodesk Maya	√	√	√
- Autodesk Motion Builder	√	×	√
- Light wave	√	√	√
- Milk shape 3d	√	×	√
- Softimage	√	×	√

- Silo 3d	√	×	√
- Google Sketch Up Pro	√	×	√
- Cinema 4D	×	√	√
- Cheetah 3D	×	√	√
- Modo	×	√	√
- Blender	×	√	√
File Supported	**GameCore**	**Unity**	**CryEngine**
- .fbx	√	√	√
- .3ds	√	√	√
- .obj	√	√	√
- .lwo	√	√	√
- .lws	√	√	√
- .ms3d	√	×	√
- .dae	×	√	√
- .dxf	×	√	√

4.3 Result of Platform Criterion

In platform criterion we compare computer platforms that each game engine can install with many platforms. The platforms that all three game engines can install are Windows, Mac Os, Wed 3d publishing. GameCore is not support Linux, Xbox, and Plat station. Unity is not support Xbox and Play station. While CryEngine can support all platforms as Windows, Mac Os, Wed 3d publishing, Linux, Xbox, and Play station. The result was showed in Table 4.3.

Table 4.3: The Result of Platform Criterion

Platform Supported	GameCore	Unity	CryEngine
- Windows	√	√	√
- Mac Os	√	√	√
- Web 3D Publishing	√	√	√
- Linux	×	√	√
- Xbox , Play Station	×	×	√

4.4 Result of Content Criterion

In content criterion we compare importing external model methods, restriction of importing custom model, effects, and saving the map with each game engines. First we compare by adding external model to each game engine. Then, we check the step to add external model and collision of game object. For GameCore, we can import external model easily. We cannot collide to the external model that we import to the map when we run the game project but can make collision by editing script of game object that we imported. We have mentioned more adding external models of GameCore already in section 3.3.4.1 of chapter 3. For Unity, we can add external model easily. We cannot collide to the external model that we import to the map when we run the game project but can make collision by generate collider in the inspector of game object that we imported. We have mentioned more adding external models of Unity already in section 3.3.4.2 of chapter 3. For CryEngine, when we want to import external model into game project we have to save the model file to folder of CryEngine which more complexity. We can collide to the external model that we import to the map when we run the game project because CryEngine editor made collision by itself. We have mentioned more adding external models of CryEngine already in section 3.3.4.3 of chapter 3.

Secondly, we compare restriction of importing custom model of each game engines. When we add game object from external folder to each editor, each of them has more restriction. GameCore, we can import custom model from external folder that we export from 3ds max in file which GameCore editor supported (see more in section 3.3.4.1 of chapter 3). Unity, we can import custom model from external folder that we export from 3ds max in file which Unity editor

supported (see more in section 3.3.4.2 of chapter 3). For CryEngine, we have to install plug in of 3ds max before exporting model from 3ds max to map of CryEngine (see more in section 3.3.4.3 of chapter 3).

Third we compare affects of each game engine which in this special project we mean special effects. For GameCore, it does not have special effects with editor. If we want the special effect we have to create animation from 3ds max or Maya like creating external model and adding it to the map like adding external model with 3d files of GameCore supported (see more in section 3.3.2.1 and 3.3.4.1 of chapter 3). For Unity, it has special effects called particle system and has nature duplication with editor which we mentioned in text 3.3.4.2 of chapter 3. For CryEngine, it has special effects and nature duplication with editor in AI function that we can use immediately which we mentioned in section 3.3.4.3 of chapter 3.

The last one of content criterion, we compare saving the game project which we want to know file that each game engines save and can open with other computer. For GameCore, the game project was saved in .wld (.world) and we cannot open game project file with other computer if we did not save all about game project in the same folder (see more in text 3.3.4.1 of chapter 3). For Unity, the game project was saved in the folder that we created in our computer and we can open the game project with other computer because Unity editor combined all about game object in our folder (see more in text 3.3.4.2 of chapter 3). For CryEngine, the game project was saved in .cry and we cannot open game project file with other computer if we did not save all about game project in the same folder (see more in text 3.3.4.3 of chapter 3). We have shown conclusions result of content criterion in Table 4.4

Table 4.4: The Result of Content Criterion.

Content	GameCore	Unity	CryEngine
Adding external model.	- Non-complexity - Make collision by editing script.	- Non-complexity - Make collision by generate collider in the inspector.	- Complexity - Make collision by editor's self.
Restrictions of importing custom model.	- Can import custom model from external folder.	- Can import custom model from external folder.	- Have to install plug in of 3d program.
Effects of game editor	- No effect with editor	- Particle system and nature duplication	- AI function
Saving the map	- Game project was saved in .wld (.world) - Cannot open game project with other computers.	- Game project was saved in the folder. - Can open the game project with other computers.	- Game project was saved in .cry - Cannot open game project with other computers.

4.5 Result of Game Play Content

In game play criterion we compare running game project, movement of game player, and error of each engines. First we compare running the game project. When we run the game project of each game engine we can see the camera view which are different each. For GameCore, when we run game project it will appear the scene that from camera of player which we specify position in the map (see more in section 3.3.5.1 of chapter 3). For Unity, when we run game project it will appear the scene that from camera of first person controller which we specify position in the map (see more in section 3.3.5.2 of chapter 3). For CryEngine, when we run game project, it will appear the scene that from camera of spawn point which we specify position in the map (see more in section 3.3.5.3 of chapter 3).

Secondly, we compare game player movement of each game engines which we will compare walking players speed of each game engines. For GameCore, the game player movement

is normal and it depends on the walking player speed that can edit in script of player (see more in text 3.3.5.1 of chapter 3). For Unity, the game player movement is normal and it depends on the player step's speed that can be edited in the inspector of first person controller (see more in section 3.3.5.2 of chapter 3). For CryEngine, the game player movement is normal and it depends on the walking player speed that CryEngine editor had specified the speed already (see more in section 3.3.5.3 of chapter 3).

The last part of game play criterion, we compare and the game's error of each game engine that will be shown in editor. For GameCore, if we make some mistakes, it will show only the error script message which we do not know where is the error and how to fix the error of game project (see more in section 3.3.5.1 of chapter3). For Unity, if we make some mistakes, it will show an error message in the lower bar of Unity editor that Unity will show the error of script and where the error happen which we cannot run the game project if we do not edit the error script before running the game project (see more in section 3.3.5.2 of chapter3). For CryEngine, if we make some mistakes in the game project CryEngine will show an error at console that will help us to find the error of game project (see more in section 3.3.5.3 of chapter3).

Table 4.5: The Result of Game play Content.

Game Play	GameCore	Unity	CryEngine
Running the game project	- Show camera view of player.	- Show camera view of first person controller.	- Show camera view of spawn point.
Movement of game player	-Depend on script of player.	- Depend on inspector of first person controller.	- Depend on speed of player in editor.
Error	- Show only error script message.	- Show error in the lower bar of editor.	- Show error at console

Chapter 5

Conclusion and Suggestion

5.1 Conclusion

Modern game engines fulfill a great set of the feature necessary for building a game project. In this special project we compare three game engines that are GameCore, Unity, and CryEngine with evaluation in five criterions such as editing, graphic, platform, content, and game play.

In editing criterion, user interface of each game engine have good design and it is easy to use for game developers. Unity is the best user interface is design because game developers can customize the interface and they can learn Unity easily. In graphics criterion, three game engines support several programs and files format. While GameCore and Unity do not support some file formats, CryEngine support all of them. Therefore CryEngine is the best between these three game engines. But the most popular programs that are 3ds max and Maya supported all three game engines so the user can export file to the format that each game engine support. In platform criterion, these three game engines can install and open the engines with many platforms. CryEngine supported all platforms that are Windows, Mac Os, Web 3D Publishing, Linux, Xbox, and Play Station. In content criterion, these three engines can add external models and each game engine have restrictions for importing the custom models. The game engines have special effect with the editor but GameCore does not have special effect with editor. In game play criterion, game developers will see the scene with camera position when they run their game project. If the game project have something wrong, the game engines will show an error and tell how to edit the error except GameCore that display only error message.

The best engine for a beginner's game developer is Unity because it is suitable with all styles of game projects such as hardcore, games in smart phone, games for children, etc. Where GameCore and CryEngine emphasize on hardcore game project.

5.2 Suggestion

This special project could be continued by comparing with some more type or new generations of game engines such as Unity 3.5 and Unreal Engine

In addition to the new generation and other game engines, the extension of this special project can be comparing the game engines with using criterions such as scripting, networking function, and game specifications.

References

[Online].Available :http://en.wikipedia.org/wiki/Game_engine

[Online].Available :http://www.flipcode.com/archives/Elements_Of_A_Game_Engine.shtml

S. Marks, J. Windsor, and B. Wunsche. "Evaluation of Game Engines for Simulated Surgical Training." **The 5th international conference on Computer graphics and interactive techniques.** New Zealand, December, 2007. pp. 273-318

[Online].Available :http://ludocraft.oulu.fi/elias/dokumentit/open_source_game_engines.pdf

[Online].Available :http://www.gamecore3d.com/

[Online].Available :http://unity3d.com/

[Online].Available :http://www.crytek.com/cryengine/cryengine3/overview

Appendices

Appendix A

Installation of GameCore

1. Download GameCore software from http://www.gamecore3d.com

Game developers have to select download link for PC version or Mac version. If they use Microsoft windows, they have to click PC version installer and if they use Apple computer, they have to click Mac version installer.

Figure A.1: Choose link which appropriate with computer.

2. Double click .exe file setup

Game developers can click next to continue or click cancel to exit GameCore setup.

3. Accept license agreement

Game developers read the license agreement. They must accept the terms of agreement before continuing installation.

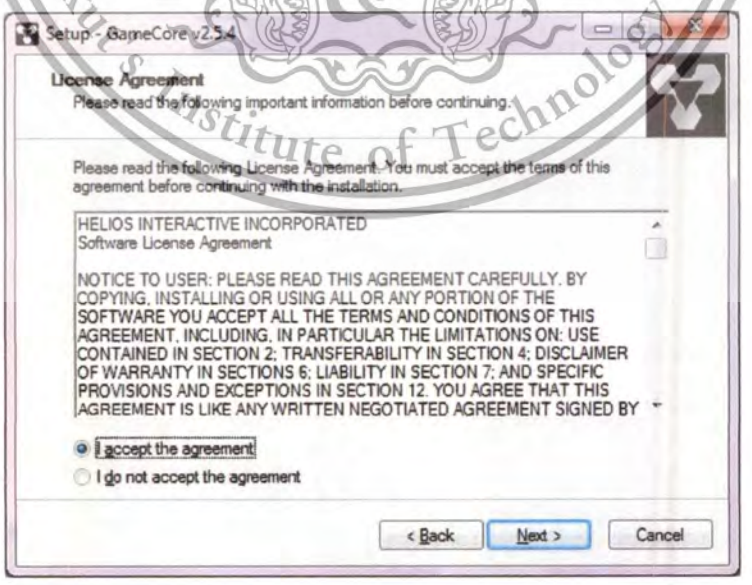

Figure A.2: Accept the GameCore license agreement.

3. Select destination location

GameCore setup will install GameCore software in to computer folder and click Next to continue. If game developers would like to select a different folder, they can click Browse.

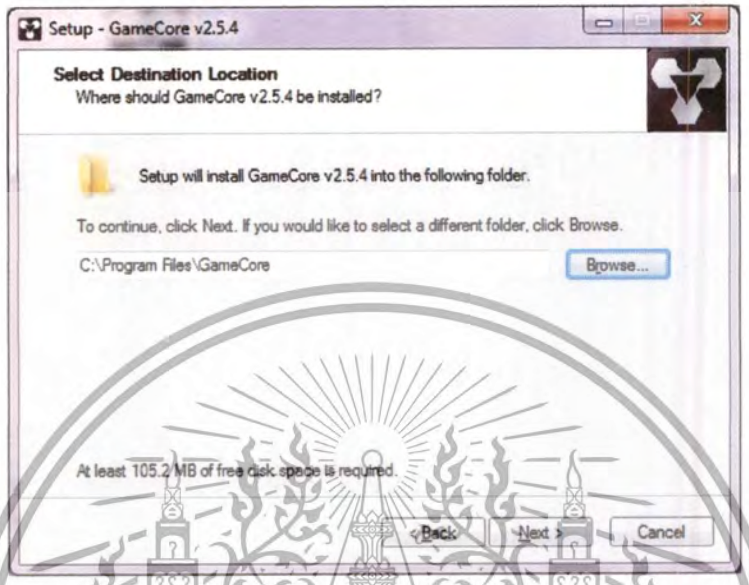

Figure A.3: Select destination location.

4. Select components

There are shown which components should be installed. Game developers can select the components that they want and clear the components that they do not want to install. Then click Next when they are ready to continue. GameCore requires disk space at least 371.9 MB.

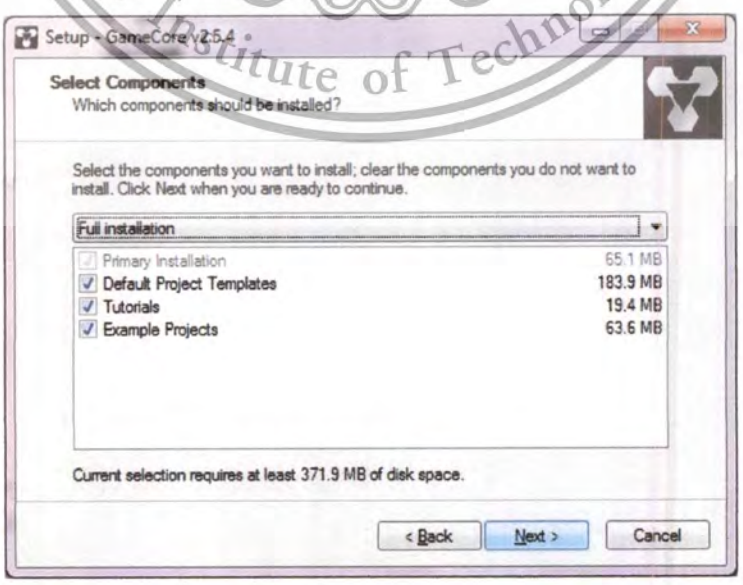

Figure A.4: Select components

5. Select start menu folder

GameCore setup will create the program's shortcuts in the start menu folder and click Next to continue. If game developers would like to select a different folder, they have to click Browse.

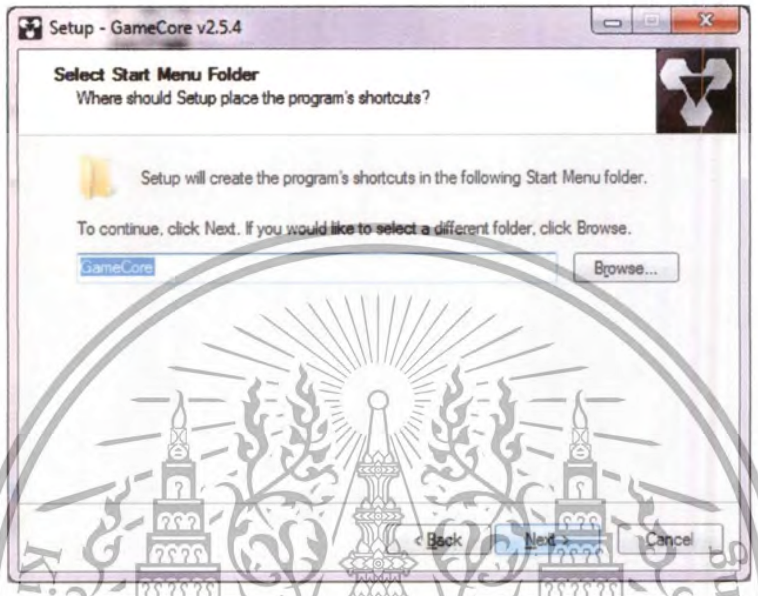

Figure A.5: Select start menu folder.

6. Ready to install

GameCore setup is ready to begin installing GameCore software on computer. Game developers have to click Install to continue with the installation or click <Back if they want to review or change any settings.

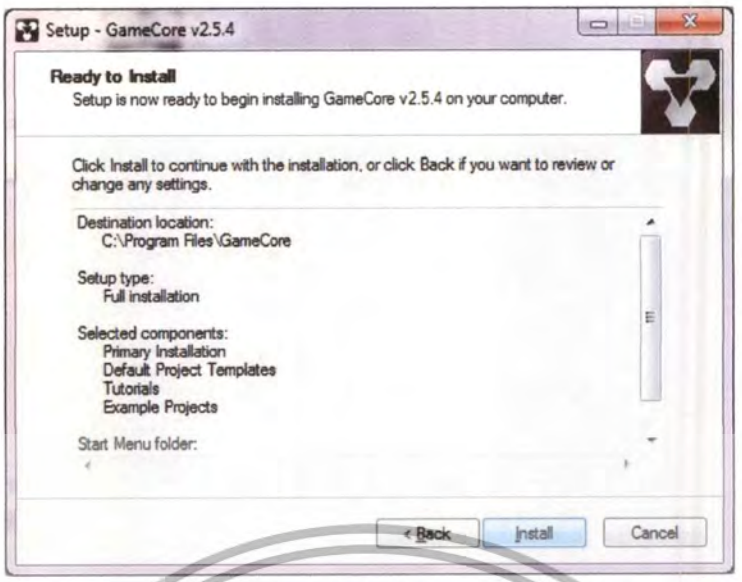

Figure A.6: Ready to install.

7. Waiting installation

Wait while GameCore setup is installing GameCore software into computer. When game developers are ready to continue with setup, they can click Next to continue.

8. Completing GameCore setup

GameCore setup has finished installing GameCore software into computer. The application may be launched by selecting the installing icons. Then click Finish to exit GameCore setup.

Figure A.7: Completing GameCore setup.

Appendix B

Installation of Unity

1. Download GameCore software from http://unity3d.com/unity/download/

Game developers have to select download link on web site. If they use Microsoft windows, they have to click Developing on Windows? and if they use Apple computer, they have to click Developing on Max OS X?.

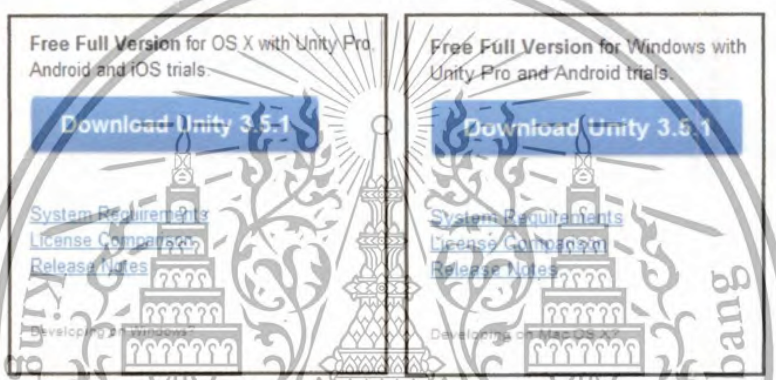

Figure B.1: Download Unity link.

2. License agreement

Game developers should review the license term before installing Unity software and press page down to see the rest of agreement. If they accept the terms of the agreement, they have to click "I Agree" to continue. They must accept the agreement to install Unity.

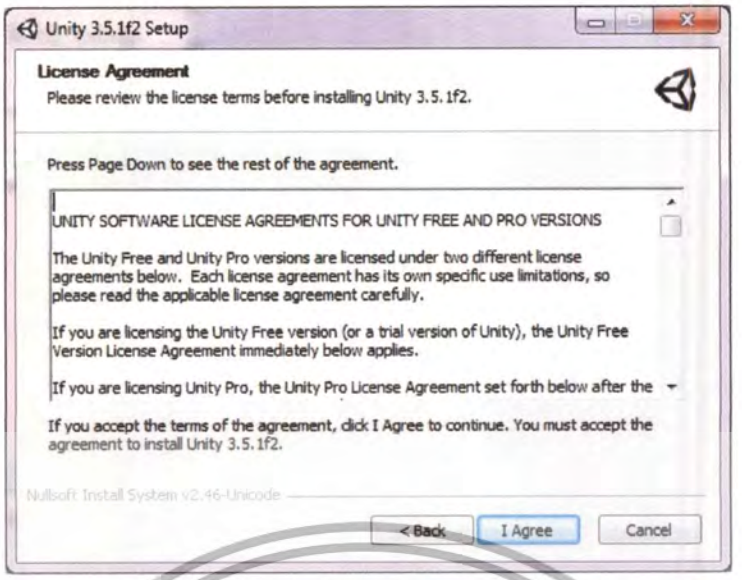

Figure B.2: License Agreement.

3. Choose components

Game developers choose which features of Unity that they want to install. They check the components that they want to install and uncheck the components that they do not want to install. Then click Next to continue setup.

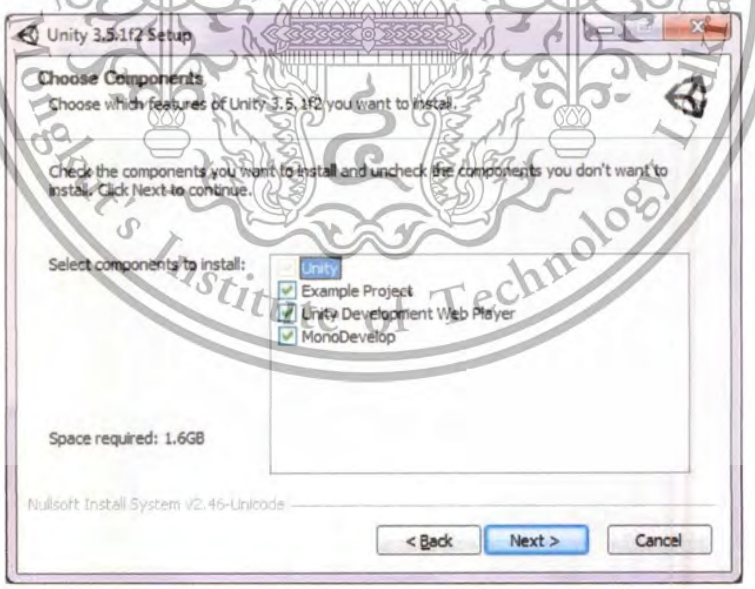

Figure B.3: Choose components.

4. Choose install location

Game developers choose folder in which to install Unity software. Unity set up will install Unity software in computer folder. If game developers would like to install in a different

folder, they must click Browse and select another folder and click Install to start installation. Unity required computer spec at least 1.68 GB.

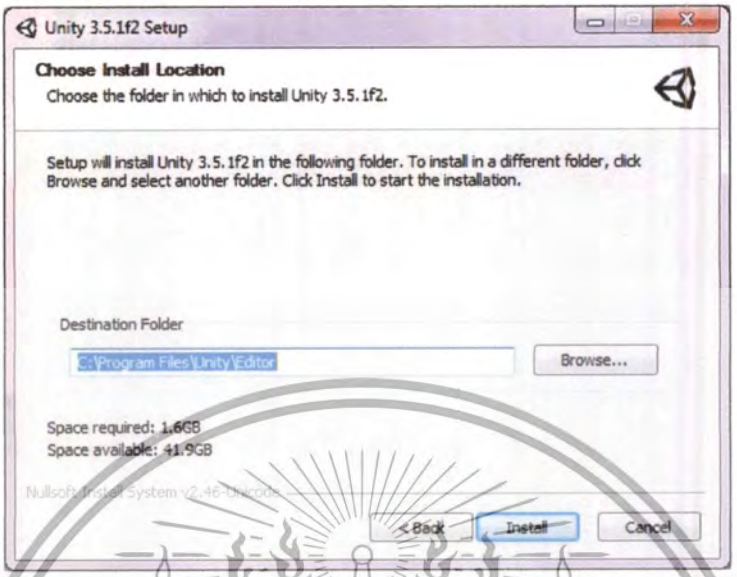

figure B.4: Choose install location.

5. Waiting installation

Wait while Unity software being installed into computer. When game developers are ready to continue with setup, they can click Next to continue.

6. Completing the Unity setup

Unity software has been installed on computer. The application may be run by selecting the installing icons. Then click Finish to exit Unity setup.

Figure B.5: Complteing the Unity setup.

Appendix C

Installation of CryEngine

1. Download CryEngine software from http://www.crydev.net/

Game developers can download a full version of the best all-in-one game development engine, for free and use it without charge for non-commercial game development.

2. Extract file .zip

Game developers have to extract file in their folder.

Figure C.1: Extract file.

3. Open folder file

Game developers opten Bin 32 for Windows 32bit or Bin 64 for Windows 64 bit.

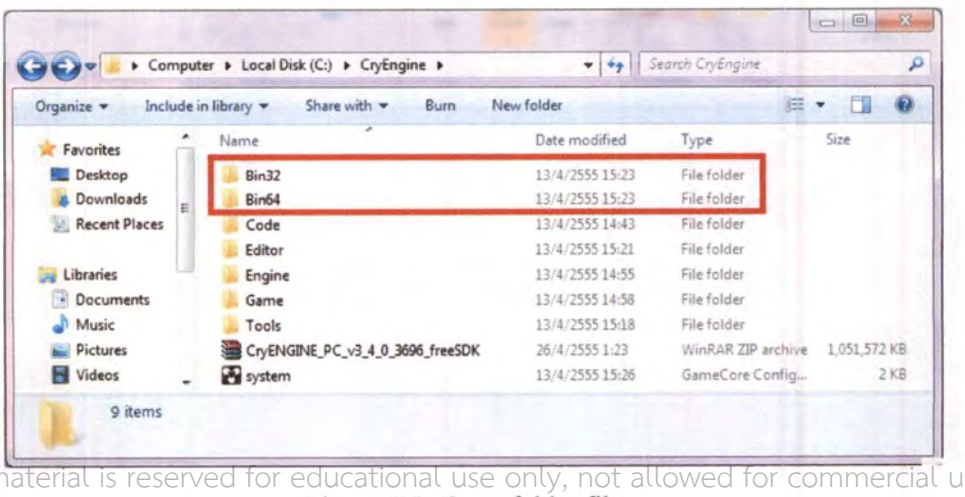

Figure C.2: Open folder file.

4. Double click .exe

Game developers double click Editor.exe to run program.

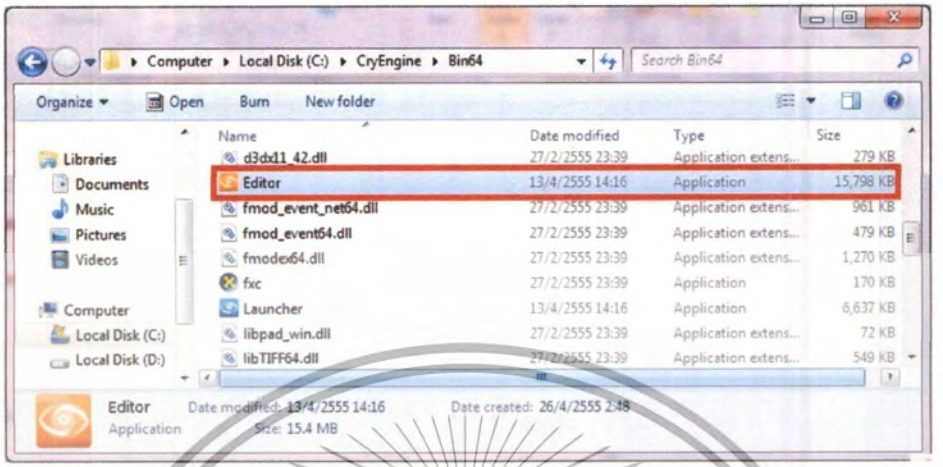

Figure C.3: Run Editor.