Design and Development of Smart Game based on Multi-Platform (Unity3D) Game Engine

AeHyun Kim, JaeHwan Bae,

1, First Author
Department of Industrial Design, Dong-A University, kah08@dau.ac.kr

*2, Corresponding Author
Department of Game Engineering, TongMyong University, bjhmail@tu.ac.kr

Abstract

This study aims to design and develop smart game based on Unity3D engine that is a multi-platform game engine. The functions that Unity3D supports autonomously are very abundant. All game developments are possible such as shader, physics engine, network, terrain manipulation, audio, video, and animation, and it is enabled so that it is possible to revise, meeting the demand of user according to the need. For the efficiency of game development process, this study aims to plan, design, and develop a smart game based on multi-platform game engine.

Keywords: Android games, Multiplatform, Smart games, Unity 3D

1. Introduction

Unity3D game engine is an integrated development tool for producing other interactive contents such as video game, architectural visualization, real-time 3D animation. Its editor runs on Window, Mac OS X, so it could make games as the platforms of Window, Mac, Wii, iPad, and iPhone. It could also produce web browser game that uses unity web player plug-in. This is a similar form of flash, and it is designed so that flash user could easily adapt even with cross domain security policy and scripting.

IMGUI (immediate mode GUI) was introduced for realizing GUI. It has an advantage that it could realize GUI with simple and small amount of script through this, but it is difficult to write GUI in complex form and event handling procedure. Currently, the version 3.1 does not support the function of writing GUI in editor and not in script.

The functions that Unity3D supports autonomously are very abundant. In fact, all game developments are possible such as shader, physics engine, network, terrain manipulation, audio, video, and animation, and it considered so that the revision is possible to the taste of user according to the need. Unity3D that produces based on Java script and C# can apply and manage after producing the desired functions with script, not producing all of the programming at once. GUI composed on screen helps the first-time developer to approach easily, and the script and program that programmer made with simple mouse drag.

This study aims to design and develop smart game based on Unity3D engine that is a multi-platform game engine. The functions that Unity3D supports autonomously are very abundant. All game developments are possible such as shader, physics engine, network, terrain manipulation, audio, video, and animation, and it is enabled so that it is possible to revise, meeting the demand of user according to the need. For the efficiency of game development process, this study aims to plan, design, and develop a smart game based on multi-platform game engine.

2. Related studies

2.1 Introduce of Unity 3D

Application using a smart Camera-Device has been remained on the level of recognition of business card, bar code and searching for words. The developments of game contents with upgraded technology will contribute to the growth of the application market. Therefore in this dissertation, we developed smart augmentation reality technology introducing recognition of a
marker, 3-D characters and character animation utilizing Camera-Devices based on Android platforms in order to develop health educational games for elementary students.

Unity (also called Unity3D) is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices, and is utilized by over one million developers.[2] Unity is primarily used to create mobile and web games, but can also deploy games to consoles or the PC. The game engine was developed in C/C++, and is able to support code written in C# or javascript. It grew from an OS X supported game development tool in 2005 to the multi-platform game engine that it is today.[2]


- Rendering: The graphics engine uses Direct3D (Windows), OpenGL (Mac, Windows, Linux), OpenGL ES (Android, iOS), and proprietary APIs (Wii). There is support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects.[4]

- Unity supports art assets and file formats from 3ds Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks and Allegorithmic Substance. These assets can be added to the game project, and managed through Unity's graphical user interface.[5]

- The ShaderLab language is used for shaders, supporting both declarative "programming" of the fixed-function pipeline and shader programs written in GLSL or Cg. A shader can include multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video card, and if none are compatible, fall back to an alternative shader that may sacrifice features for performance.[6] Unity also has built-in support for Nvidia's (formerly Ageia's) PhysX physics engine, (as of Unity 3.0) with added support for real-time cloth simulation on arbitrary and skinned meshes, thick ray casts, and collision layers. [7]

- Scripting: The game engine's scripting is built on Mono, the open-source implementation of the .NET Framework. Programmers can use UnityScript (a custom language with ECMAScript-inspired syntax), C# or Boo (which has a Python-inspired syntax).[8] Starting with the 3.0 release, Unity ships with a customized version of MonoDevelop for debugging scripts.[9]

- Asset Tracking: Unity also includes the Unity Asset Server - a version control solution for the developer's game assets and scripts. It uses PostgreSQL as a backend, an audio system built on the FMOD library (with ability to playback Ogg Vorbis compressed audio), video playback using the Theora codec, a terrain and vegetation engine (which supports tree billboarding, Occlusion Culling with Umbra), built-in lightmapping and global illumination with Beast, multiplayer networking using RakNet, and built-in pathfinding navigation meshes.[10]

- Platforms: Unity supports deployment to multiple platforms. Within a project, developers have control over delivery to mobile devices, web browsers, desktops, and consoles.[11] Unity also allows specification of texture compression and resolution settings for each platform the game supports.[11]

Currently supported platforms include Windows, Linux, Mac, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, and Wii. Although not officially confirmed, Unity also supports the PlayStation Vita as can be seen on the game Escape Plan. Upcoming platforms include BlackBerry 10, Wii U, Windows 8, and Windows Phone 8.

- Asset Store: Launched in November 2010, the Unity Asset Store is a resource available within the Unity editor. The store consists of a collection of over 4,400 asset packages, including 3D models, textures and materials, particle systems, music and sound effects, tutorials and projects, scripting packages, editor extensions and online services.
The store also contains many extensions, tools, and asset packages such as the package NGUI: Next-Gen UI by Tasharen Entertainment,[12] and the visual scripting extension uScript by Detox Studios, Tidy Tile Mapper,[13] a 2D/3D tile-based game design extension by Doppler Interactive, and the input scripting package FingerGestures.

- **Versions:** The first version of Unity was launched at Apple’s Worldwide Developers Conference in 2005. It was built to function and build projects on Mac computers and garnered enough success to continue development of the engine and tools for other platforms.[2] Unity 3 was released in September 2010 and focused on introducing more of the tools that high-end studios have at their disposal. This allowed the company to capture the interest of bigger developers while providing independent and smaller teams with a game engine in one affordable package. The latest version of Unity, Unity 4.0, was released in late 2012, and includes additions such as Mecanim animation and DirectX 11 support.

### 2.2 3DS MAX Characteristics

Modeler and texture artist could increase the speed of working and efficiency because of extensive polygon modeling and texture mapping tool set. CAT (Character Animation Toolkit) that is completely integrated provides advanced rigging and animation system that can be used immediately. Artist could create very realistic or great image by using mental ray renderer and integrated 3ds Max scan line including mental ray network that can finish rendering faster. Since it supports C++ and .NET, developer could customize and expand 3ds Max and also integrate it into the existing pipeline. Also, developer could create 3D model with Autodesk 3ds Max, send it to Autodesk FBX exchange technology file after animating it, and then bringing it to Unreal Editor contents production software [1].

### 2.3 Technical Importing 3DS MAX data from Unity3D

Please. The things that are imported from 3DS MAX from Unity3D can be largely divided into five parts. First, it is every node that has position, rotation, and scale. Second part includes pivot point, names, vertex, and meshes with colors. Third, they are normals and one or two UV sets. Fourth, they are diffuse texture and materials with colors. Fifth, they are many materials per mesh, animation, and bone based animations. Thus, these five parts are the ones that Unity recognizes from 3DS MAX. Also, Unity3D could import FBX file. FBX provides supports for software and applications that are both independent and from certain companies. FBX is used within entertainment pipeline or as a part of design production. Files can be sent more smoothly, more data is maintained, and the work flow is more efficient. To send a file from 3DS MAX to Unity3D, first convert to FBX file with export from 3DS MAX, and send it. Then the file can be imported from Unity3D. Only, if the name of FBX file is in Korean, Unity3D cannot recognize it [1][2].

### 3. Body

#### 3.1 Summary of Game Development

In this research
1) Game Name: SPARDDA
2) Game Genre
   - Platform: supports smart multi-platform
   - Genre: action game (sound action game)
3) Game Characteristic
   - Subject
     Battle of Thermopylae in Greece Sparta as the background, it is a game of defending against Persian soldiers that attack in a narrow canyon.
   - Method of playing by utilizing the touch function that is the advantage of touch phone and tablet
Similar to the existing rhythm game, method of defeating the enemy by listening to the sound of the enemy and responding to it

4) Core Idea within the Game
▷ Escaping the general frame of action game that move, attack, defend, and avoid by controlling the existing occupation characters, users must immediately respond to the sound. It is a speedy and thrilling game that requires high concentration even with simple control.
▷ Enemies randomly attack with previously recorded “S!” “Pa!” “R!” “Ta!” or other special voices, and users must defend the enemies with two methods of responding to the short shout of the enemies through touch gesture or pushing buttons, not gestures, to the special voices.
▷ When it is hard to hold the enemies’ attack, users can use items purchased from a store within the game or deadly technique to escape from a crisis, which can be used when the action gauge becomes full after succeeding an attack.

The platform of the game is android smart tablet base, and the genre is sound action game. It is designed with the Battle of Thermopylae in Greece Sparta as the material.
3.2 Graphic Concept Art

<table>
<thead>
<tr>
<th>Table 1. Original Picture Design of Character</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
<td><strong>Image</strong></td>
</tr>
<tr>
<td>After looking at the proposal, diverse concept data was researched and 2D work was done by using Photoshop, and concept was drawn so that it would be easy to understand when 3D modeling.</td>
<td><img src="image1.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Modeling Work</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
<td><strong>Image</strong></td>
</tr>
<tr>
<td>Based on the 2D concept, low polygon modeling work was done by using 3D MAX, and in order to prevent the phenomena of a side getting crushed when the movement is given early at the time of basic modeling, it was modeled, considering a proper division of the side.</td>
<td><img src="image2.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3. UVunwrap</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
<td><strong>Image</strong></td>
</tr>
<tr>
<td>When working on UV after creating a low polygon model from 3D MAX, the source of check pattern was put, and UV were organized according to the scale of UV, and the number of the map was decreased as much as possible. At the time of UV work, UVmaster from ZBRUSH and UVunwrap from 3D MAX were used to shorten the time of UV work.</td>
<td><img src="image3.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4. Mapping</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
<td><strong>Image</strong></td>
</tr>
<tr>
<td>After putting in basic colors and materials data by using Photoshop through UV data, the speed of work was increased through the real-time painting through Bodypaint 3D.</td>
<td><img src="image4.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5. Animation</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
<td><strong>Image</strong></td>
</tr>
</tbody>
</table>
| |!
After completing the modeling and mapping, conduct the work of giving physique to the vertex of the character through biped and attach the modeling appropriately to the biped. Afterwards, adjust the animation keys with the auto key and put in the motions appropriate to the plan that has been set up.

<table>
<thead>
<tr>
<th>Table 6. Corrections and Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
</tr>
<tr>
<td>By putting animation, make corrections such as adjusting how distorted the joint part becomes and how much the mapping is stretched. When some part is insufficient, add modeling or mapping and apply animation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7. FBX File Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
</tr>
<tr>
<td>When the work is complete, first, adjust the size in the Units setup and put in the setup for the FBX file through Export. And then take out the camera and light, set up the frame interval in the animation part, save it, and apply it to the unity.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 8. Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description on Graphic Design Work</strong></td>
</tr>
<tr>
<td>Undertaking the interface design of each button, design according to the flow that is appropriate to the entire frame. Usually, the design and stones in the background of ancient Greece were used for expression. The designs for the times when the button is pressed and when it is not are inserted separately. The designs for items that can be used on the game screen and various designs for the standby screen are composed.</td>
</tr>
</tbody>
</table>

3.3 Subsequent Game Development

3.3.1 Game UI Screen

1. **Scenario**
   It is the very first menu that begins the game, and when beginning, it shows the story and instruction for how to play the game. User can select the difficulty of gameplay (Easy, Normal, and Hard) and play the game.

2. **Custom**
   It is a mode that enables user to enjoy different mode other than the scenario mode (Defense, Push, Rush, and Marathon), and user can enjoy diverse event games (currently, yet to be materialized).

3. **Shop**
   User can purchase items that can be helpful to gameplay. Currently, only consumable items are materialized.

4. **Option**
This is the place where user sets up the options, and user can set touch sensitivity and sound and choose whether to skip the story when the scenario is playing.

⑤ Exit
Confirm whether to end the game or not.

### 3.3.2 Gameplay

▷ Currently, only the scenario mode is materialized, and it is planned to materialize the custom mode in future.
▷ Before the gameplay, a short story and instruction on how to play game appear. User can move onto the next page by touching the screen, and the game continues.

Select the difficulty of scenario mode -> Scenario progress screen -> Explanation on how to continue the game -> Screen of easy gameplay of scenario -> Menu screen within the game -> The screen of game ending (Victory or Defeat)

▷ It shows all the explanation on the game interface, and when user plays it for the first time, it is shown at the end of the scenario. When beginning from the option window, user can check the indications.
▷ In the window above, if user touches one more time, then it shows the instruction related to the game on the window that has appeared. Similarly, it can be set in the option window, and when touching the screen, it moves on to the beginning of the game, and the loading screen that loads the game is created in the middle.
4. Conclusion and Proposal

This study aims to design and develop a smart game based on Unity3D engine, which is a multi-platform game engine. The functions that Unity3D supports autonomously are very abundant. All game developments are possible such as shader, physics engine, network, terrain manipulation, audio, video, and animation, and it is enabled so that it is possible to revise, meeting demand of user according to the need. For the efficiency of game development process, this study aims to plan, design, and develop a smart game based on multi-platform game engine.

5. References