The Implementation of a 3D Game Engine based on DirectX 9

Hyun Myung Kang
Div. of Multimedia Engineering
Hanbat National University, Daejeon, Korea

Woo Seop Rhee
Div. of Multimedia Engineering
Hanbat National University, Daejeon, Korea

ABSTRACT

Recently, almost games are using the 3D environment. Therefore, it required strongly that well-structured 3D engine or tools for development of some complicate 3D applications efficiently. In this paper, we design and implement a 3D engine (PLay engine) using the DirectX 9 SDK of the Microsoft corporation. The PLay engine has independent module structure, which has object oriented characteristics, and has not only 3D rendering functions but efficient algorithms. Moreover, we implement some tools what has compatibility with our engine for convenience. Therefore, it helps development of a 3D application easily and efficiently. We also describe each module with 2-layer structure, and each tool with compatible module, and make a simple game using PLay engine for testify.

Keywords: *Game Engine, 3D Graphics, DirectX, Game Programming, Game Algorithm*

1. INTRODUCTION

Recently, almost games are presented by the visual environments using the 3D graphics. Console games, package games and also online games are with no exception. There are many user groups which support that environment. For the good examples, 'Lineage2' of NC soft, 'Doom3' of ID soft and 'Gears of War' of Epic are existed.

3D graphics technology as well as user's demands has been contributed in the 3D game development. In hardware, the graphic acceleration card technique grew rapidly with the GPU (Graphics Processing Unit) what can process high speed rendering. In software, the API (Application Programming Interface) or rendering optimization algorithms are researched for the best hardware performance. Programmers can implement the 3D games using the API and the optimization algorithms. These know-hows and techniques are used to make a game, and used to make some other games by supplement. Therefore, these know-hows and techniques are managed by encapsulation like a software engine. This is a game engine and has many advantages. So, almost game companies develop it themselves or buy a testified commercial game engine [1],[2],[3]. But commercial game engines are too expensive and dependent on its target game. If the open source game engines are exist, it is also dependent on its own libraries. Therefore, it is hard to generalization.

In this paper, we implement the game engine using DirectX 9 that use by many popular game engines base on the low level API. The implemented game engine is generalized by use of API directly and eliminated library dependencies except the DirectX. Therefore, it can be not only a basis for game development but for integration with other engine easily. Moreover, it can be a basis of other game engine.

The rest of this paper is organized as follows. Section 2 describes background of general 3D game engines, section 3 describes implementation and the game engine's layers/modules, section 4 explains about experimentation and short conclusion remark.

2. BACKGROUND

2.1 DirectX SDK

The DirectX SDK (Software Development Kit) is Microsoft foundation software development kit, Windows OS based API made by C++, for the 3D application development [4],[5]. As it has built-in communication techniques to use a 3D graphic acceleration card directly, a programmer can access to hardware by function call easily according to existing rules. The DirectX SDK also provides many classes and utility

* *Corresponding author. E-mail : wsrhee@hanbat.ac.kr*
Manuscript received Aug. 20, 2008 ; accepted Sep. 22, 2008

functions to implement 3D, sound, I/O and network. It can be used for basis of library efficiently in an application programs.

2.2 General Requests for Game Engine

The general required functions for Game engine are follows [1],[3]:
- Access to rendering pipeline for represent 3D
- Easily use for mathematical model used in 3D graphics
- Management and easy mesh loading,
- Outer/inner terrain
- Management of animation character
- Collision detect
- Sound, special effect, physics modeling, etc.

And, implementation with optimized algorithms for real-time rendering is also requested. In other words it should be implemented with least loss quality and most operation speed [1],[6],[7].

3. IMPLEMENTATION

The game engine implemented in this paper include embed the algorithms that are guaranteed performance and used generally. And it composed the independent modules according to the functions and never has long program than need be due to include necessary module. Also, it has scalability and reusability due to implemented with layered structure. We named our engine "Pickable & Layered 3D engine" (PLay engine). PLay engine composed of lower based modules (1st layer), special function extended modules with integrated those modules (2nd layer), and simple tools that provides the user interface can use those modules easily (interface tool) as shown in the figure 1. The base modules consist the basis layer (1st layer) and extension layer (2nd layer) is on the basis layer, 2-layer structure. Therefore, it can extend easily by redesigning of 2nd layer as occasion demands.

3.1 First Layer

First layer has basis abilities of the engine. That is, rendering, I/O, sound modules were isolated. Therefore, it can use not only games but any application programs, and it has good scalability and reusability due to include just necessary modules.

3.1.1 Direct3D(D3D) Module

The Direct3D module manages the rendering pipeline which is the basic function for representing of 3D. The Direct3D9 and Direct3DDevice9 are interfaces of these functions in the DirectX [5]. So, the Direct3D module manages these two classes. It also manages the Material, Light and the Display mode automatically, and can communicate with devices for advanced setting by the user.

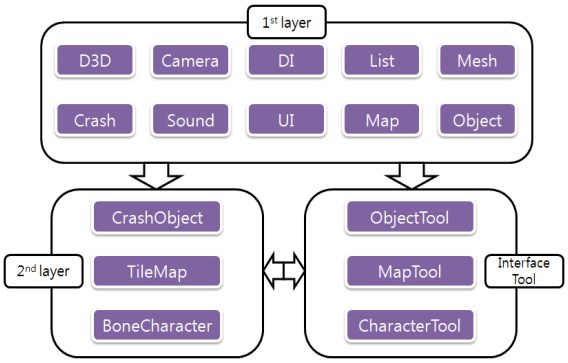

Fig. 1. Structure diagram of the PLay engine.

3.1.2 Camera Module

The camera module manages View and Projection transforms of the 3D graphics. So, user can use this module easily like camera control. It also optimized by minimized necessary calculation due to check on the camera's movement.

3.1.3 DirectInput(DI) Module / Sound Module

The DirectInput module provides correspondence with keyboard/mouse by the DirectInput interface.
The sound module can manage simple sound files like wav, midi, based on the DirectMusic interface.

3.1.4 List Module

The list module is made for use the data structures easily. It is quietly the light module due to the independent module, different from STL (Standard Template Library). It provides a choice of dynamic/static structures, the integration of array/stack/queue functions flexibly.

3.1.5 Mesh Module

The mesh module provides loading and management function of 3D models. The figure 2 shows an example of the 3D object, designed through the 3DS MAX, loading and rendering by the mesh module.

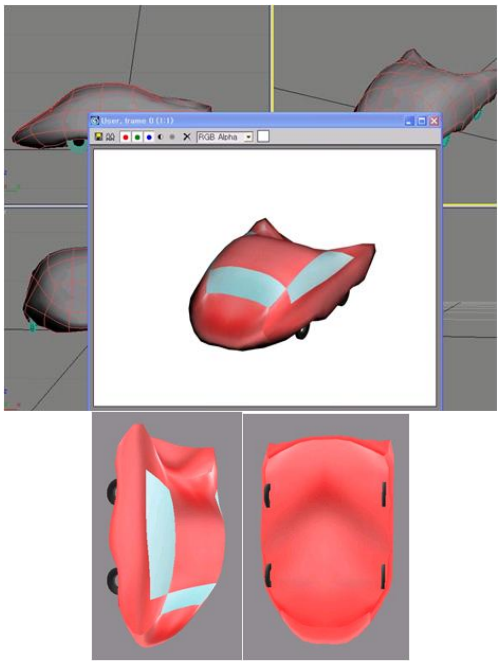

Fig. 2. Make a mesh with 3DS MAX and loading/rendering by the mesh module.

3.1.6 Object Module

The object module manages the special object's conceptional position, rotation, scale information easily. It supports interact with other mesh and the mesh module of the PLay engine.

3.1.7 Map Module

The map module is the outdoor terrain engine and manages the outdoor terrain totally. It creates a map using the height texture (height map) basically. It also includes the LOD (Level Of Detail) method that controls the polygon details according to distance based on the camera position and the Quad-tree frustum culling method that enhance the rendering speed without quality loss due to exclusion of the external region view frustum in the rendering using the map block [8],[9].

When we draw a sky, it can be large load to rendering a huge sky, so usually a trick is used. This module provides the skybox trick method [10]. The figure 3 shows examples of rendering terrain with sky and optimized rendering by culling.

3.1.8 UI Module:

The UI module provides the function of a general user interface. It helps drawing letters or 2D sprites on the screen and manages this easily. The figure 4 shows an examples of drawing fonts on a sprite and transformed sprite using the UI module.

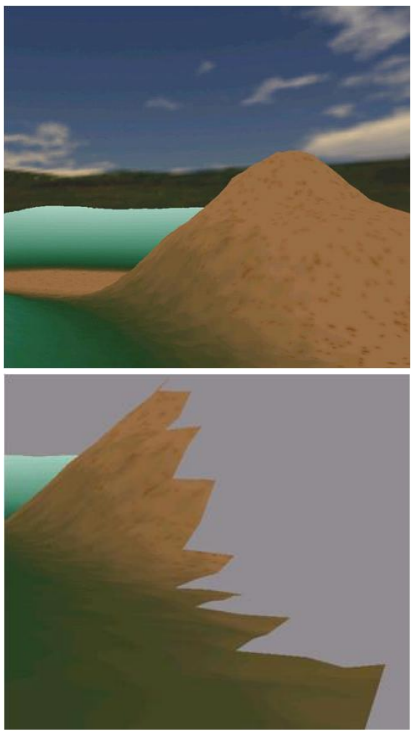

Fig. 3. A created map by the map module, a part of optimized map by culling.

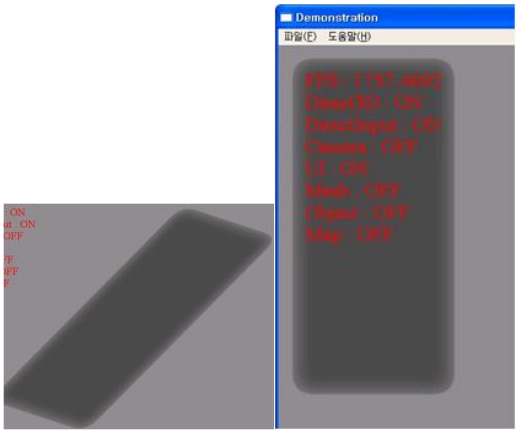

Fig. 4. Draw transformed sprite and fonts.

3.1.9 Crash Module

The crash module provides the functions of collision detection. It defines various collision objects and helps the collision detection of them.

This module has the BS (Bounding Sphere), the AABB (Axis Aligned Bounding Box), the OBB(Oriented Bounding Box) basically. It can use the definition of additional collision object by inheritance of this module and the collision formula between each other. This module use the optimized collision detection algorithm for OBB called "Fast Overlap Test for OBBs" [11].

3.2 Second Layer / Tools

The second layer is a more powerful layer than the first layer that provides a basic function by the modification/integration of the first layer's modules as occasion demands. The second layer is reusable in the other application program but more restrictive than the first layer.

We made the tools for integration from the first layer to the second layer easily and directly. It can make with visual interface and can load in the second layer by the self defined exporting data format. These tools support the enhancement of engine capability efficiently.

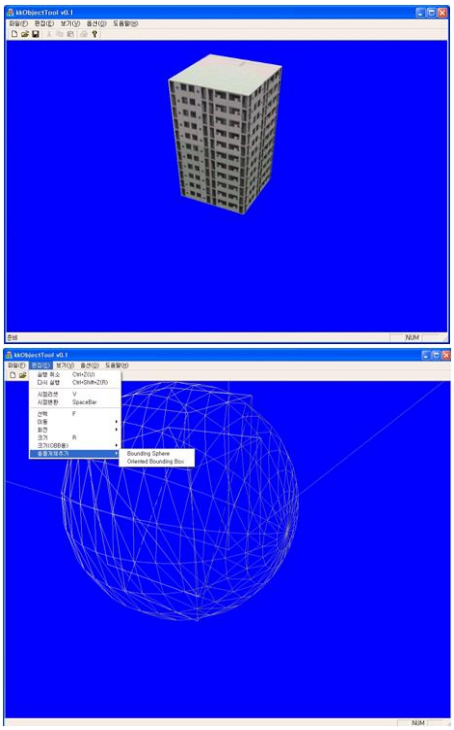

Fig. 5. Loading and processing a mesh by the object tool.

3.2.1 Object Tool

The object tool integrates mesh/object/crash modules and provides the object that can apply the collision processing. It also supports visual interfaces that can make basic size, rotation, central point of object. The figure 5 shows an example scene of loading a mesh and link up the building object with the BS and the OBB by the object tool.

3.2.2 CrashObject Module

The CrashObject module defines a crashable object and can load the files made by the object tool. The process of collision detection processing is constituted two tests, the BS is first and the OBB is second. The BS is used to decide a necessity of collision test due to the fastest collision test, and the OBB decides the real collision test. This module includes the Mesh Container internally and manages the mesh data automatically since it needs only once loading of the redundant mesh data.

3.2.3 Map Tool

The map tool makes the real map through the integration of the Map/CrashObject modules and indicates the on-map objects. The map is composed of tile type using the texture mapping method on the designated tile. And it can place objects on the map voluntary. The figure 6 shows examples of a tile map and place on-map objects by the map tool.

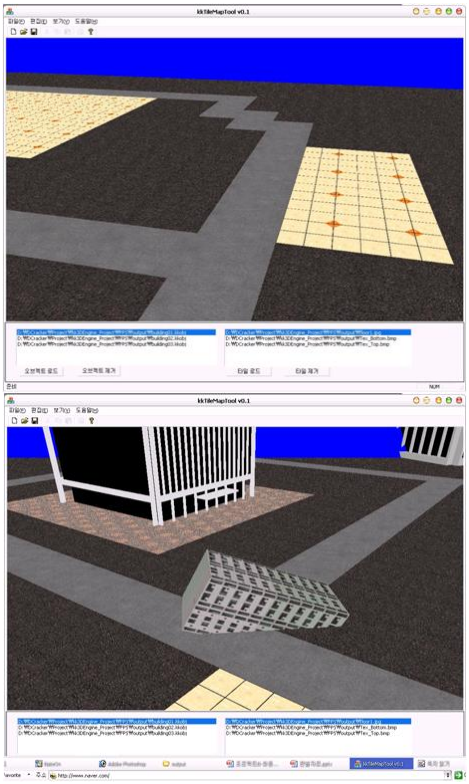

Fig. 6. Make a tile map and place on-map objects by the map tool.

3.2.4 TileMap Module

The TileMap module defines the map based on the tile type and can load the files made by the map tool. This module optimizes the on-map objects through the Quad-tree culling technique provided by the map module. And it also manages and optimizes the objects on the map internally using the same technique. This module includes the TileTexture Container inside and manages automatically once loading to the redundant texture data.

3.2.5 Character Tool

The character tool provides the hierarchy character processing interface based on Bone. It can process the realtime-test with forward/backward play of animation and can link collision object to each special part of character. The figure 7

shows examples of the animation of a hierarchical character and scene of link the collision object to each part with visual interface by the character tool.

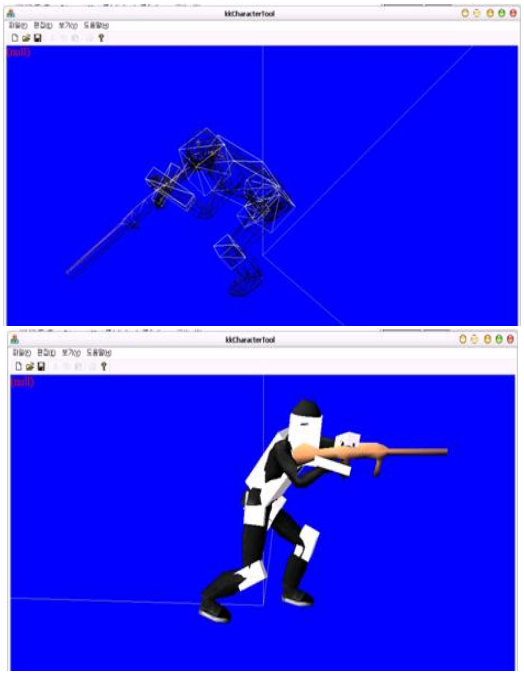

Fig. 7. Load a hierarchical character, link collision object to each part by the character tool.

3.2.6 BoneCharacter Module

The BoneCharacter module defines a hierarchical animation character and can load files made by the character tool. This module manages the current states and provides the play function of animation in the special frames according to the state. It also provides the animation controller with Play, Pause or Rewind, and can support handling the character as an object by inheritance from the object module. The BoneCharacter module can also detect the detail collision region. (etc. Head, Etc1 or Etc2)

4. CONCLUSION

In this paper, we implemented the PLay engine that has a good scalability and independency. The PLay engine is easy to use due to the independent composition modules. It also has good points in the view point of extension or combination with other game engine. And these merits make decreased time of development when we implement the real game.

Moreover, the shader, as a GPU's programming language, is required essentially and provides the powerful function recently. Therefore, game engines should deploy the shader easily according to these trends. The PLay engine can not apply the shader due to usage of basic function. But, it can upgrade to apply the shader through the inheritance and redefinition due to the object oriented design. However, the separated shader management module is better than the inheritance and redefinition for efficiency and scalability. For that reason, future studies are needed to achieve a game engine full-supports the independent shader module for easily usage without upgrade.

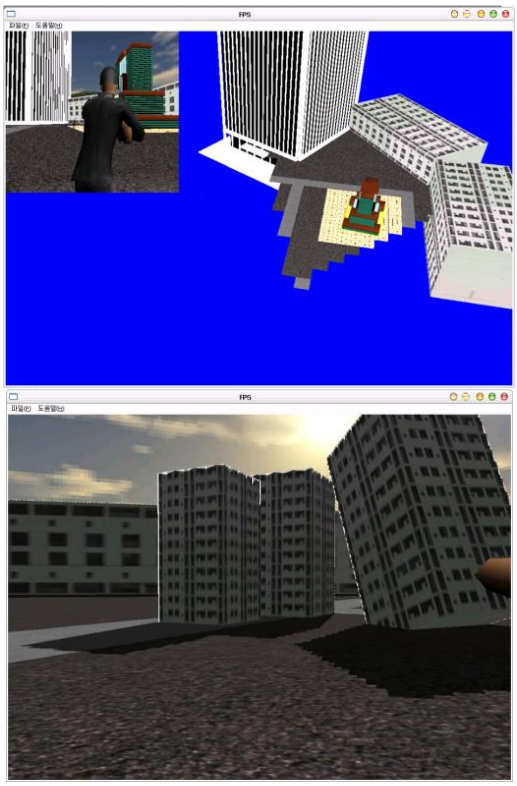

Fig. 8. A scene of the PLay engine test, apply depth shadow shader to the PLay engine.

The figure 8 shows the screen shot of the FPS (First Person Shooting) game based on the PLay engine and apply shader for depth shadow technique by upgrading.

REFERENCES

[1] S. H. Kim and T. J. Park, "Proposal of Game Contents and Game Engine Technology (Korean)," Journal of the Korea Multimedia Society Special Edition, *vol. 8, no.* 1, Mar. 2004, pp. 1-15.

[2] K. B. Lee and J. Hwang, "A Study of Game Technology and Trend (Korean)," Journal of The Korea Multimedia Society Special Edition, *vol. 9, no. 2*, Jun. 2005, pp. 1-7.

[3] D. H. Eberly, 3D Game Engine Design: A Partical Approach to Real-Time Computer Graphics, *Morgan Kaufmann Publishers*, California, 2001.

[4] http://www.microsoft.com/korea/windows/directx/productinfo/overview/default.mspx.

[5] F. D. Luna, Introduction to 3D Game Programming with DirectX 9.0, *WordWare Pub*., Texas, 2004.

[6] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," *SIGGRAPH 1993*, 1993, pp. 19-26.
[7] H. Hoppe, "Progressive Mesh," *SIGGRAPH 1996*, 1996, pp. 99-108.
[8] Y. J. Kim, 3D Game Programming, *Hanbit Media Inc.*, Korea, 2007.
[9] M. de Berg and K. T. G. Dobrindt, "On Levels of Detail in Terrains," Graphical Models and Image Processing, vol. 60, no. 1, Jan. 1998, pp. 1-12.
[10] J. R. Isidoro and P. V. Sander, "Animated Skybox Rendering and Lighting Techniques," *SIGGRAPH 2006*, 2006, pp. 19-22.
[11] http://user.chollian.net/~manilee/Fast_Overlap_Test_for_OBBs.pdf.

Hyun Myung Kang
He received the B.S. in the department of Multimedia Engineering from Hanbat National University, Daejeon, Korea in 2008. And he is currently a M.S. student in the department of Multimedia Engineering at Hanbat National University. His research interests are concerned with broadband network architecture, quality of service in Internet and mobility management with multicast.

Woo Seop Rhee
He received B.S. degree in Computer Science from Hong Ik University, Seoul, Korea, in 1983 and M.S., and Ph.D. degree in 1995 and 2003, respectively, in Computer Science from Chungnam National University, Daejeon, Korea.
From 1983 to 2005, he was with the Electronics and Telecommunication Research Institute (ETRI). He involved in development of TDX switching system, HANbit ACE ATM switching system and Optical access system as a project leader.
In 2005, He joined Hanbat National University, Daejeon, Korea and is currently assistant professor of Multimedia engineering department. His research interests are concerned with broadband network architecture, quality of service in Internet and mobility management with multicast. He is an active member of ITU-T SG 13 as Editor and member of KICS, KOCON in Korea and IEEE.