Online and Mobile Cross Platform Technology for using 3D Resources on 2D Mobile Systems using a CM3D Engine

Sungkwan Kang[1], Joonseub Cha[2], Jimin Lee[1] and Jongan Park[1]

[1] Chosun University, [2]Honam University
arome@chosun.ac.kr

Abstract

These days, game and contents have been served in PC, game consoles, smart phones, and media system and interconnection between platforms of such systems is an important task to be resolved. Although the development of systems to support 3D is increasing, a considerable amount of 2D-based system has been still used. As existing 3D engines are optimized for the 3D supporting system, there is a difficulty in executing contents in the only 2D-based system. Also, we have to develop online and mobile cross platform technology to compensate for lack of the content elements while sharing the benefits of online contents and mobile contents. This study developed a CM3D engine (Cross Mobile 3D), online and mobile sharing platform system based on a software renderer which can execute 3D content on a 2D mobile system. The CM3D engine was designed to access 3D content in the systems which normally only support 2D content. Following this, functionality was added to support the reuse of resources through automatic re-processing of existing PC-based content for the online and mobile use. As a result of tests using a 2D mobile phone which does not support 3D, it was discovered that the CM3D engine has similar performance to an existing 3D engine.

Keywords: *3D, Contents, Mobile, Game, Platform, online/mobile interconnection*

1. Introduction

Recently, contents have been realized with independent forms in the platform and as the importance of 3D contents is rapidly increasing, relevant studies have been performed. However, only 2D supporting systems are still used in many areas and the need to develop technology to support 3D contents on a 2D-based system is increasing.

For displaying 3D contents, mobile systems can't work smoothly because of differences and limitations in systems in comparison with PC based systems, and mobile systems usually function only through the use of 3D hardware chip-sets and expensive 3D software engines. Therefore, 3D engine technology which can support 3D contents in cheap terminals and work with contents in other systems such as mobile terminals or PCs is necessary. Also, along with the decreased differences in mobile platforms due to expansion of mobile markets and convergence of platforms, increasing demand of 3D contents for 3D mobile games and interconnection of online games with 3D contents, 3D contents has become an essential issue for future mobile markets [1].

Because of this, major communication companies are leading in the development of standard technology for a mobile 3D environment and the most important area in the development of mobile 3D technology is the standardization of the technology. Now, Khronos group [2] including the Nokia, Ericsson, Motorola and the JSR 184 group [3] led by Sun Microsystems are active in enacting API standards in the mobile 3D field [4]. The most

representative mobile 3D engine now available in the market is the G3 engine [5] for Gomid mobile phones.

The G3 engine is a 3D engine for CDMA type 3G mobile phones and consists of a 3D engine, a 3D avatar system, a 3D world viewer, and a 3D authoring tool to create 3D contents. The size of the engine is about 500KB. This engine performance supports to drawing 20 frames a second in a 500 polygon environment.

Then there is the M3D [6] engine by Reakosys. It is a 3D engine which supports both CDMA and GSM, and works on 2.5G or 3G terminals and supports various mobile platforms such as Brew, J2ME, WITOP and GVM. The application using this engine is about 20MB in size and engine performance depends on the size of the graphics data, but it supports from 7 to 10 frames a second.

Mascot Capsule engine [7] by HI Cooperation in Japan, is a 3D engine which had a good reputation when it was embedded in a DoCoMo 505i by NTT, Japan. The engine size is about 50KB and the application made using the engine is about 1MB in size.

It provides various 3D rendering and special effects, and there are 2 versions a CAPI type engine and a Java API type engine. It provides rendering of 20 frames a second in a 500 polygon environment. As there is also a PC or Web version, it is possible to make contents under the concept of One Source Multi Use [8].

Lee and others [9] proposed a 3D geometrically transformed engine structure which can be applied to 3D mobile graphics processors, and Lee and others[10] designed pipelines whose stages can change variably to minimize the latency period of each lighting effect and the energy required.

Also, Ted Zuvich [11] presented a simplified simulation model for a racing game and Shin and others [12] proposed a context-aware component which is aware of physical elements occurring when an automobile is driven in a game where a mobile physical engine is applied and decides on its own behavior itself. As mobile systems with cubic display panels have appeared, more attention has been paid to making more absorptive contents.

Song, *et al.*, [13] conducted a study on frame standardization which includes a 3D GUI widget and a toolkit for which the technologies of a 3D platform content engine and an application frame are combined and Lee [14] proposed a structure for a square root and inverse square root processor which can be applied to lighting engines and shader processors for mobile-based 3D graphics processing.

In particular, a number of studies have been conducted for better performance and optimalization using OpenGL ES, a mobile engine based on OpenGL library. BaeK[15] proposed how to effectively realize OpenGL ES 1.1 Standard in environment where the function of OpenGL is provided as an exclusive hardware such as desktop and Lee[16] proposed new commands that efficiently adopt delta time and unit structure from the GL ES Standards. Also, Pulli [17] reported how OpenGL ES can obtain optimal properties and performance in different systems and how graphic structures in PCs can be connected with interfaces. Chenpp [18] proposed new framework for developing 3D graphic applications on portable devices such as cellular phones or personal digital assistants.

Therefore, this study designed a 3D renderer and support library for 2D mobile systems, which can easily work 3D contents in 2D based mobile systems which currently have a large market share. It focused on realizing a cross platform which considers connection between other systems (PC, Mobile Phone, PDA, Console game device) for compatibility of existing OpenGL and OpenGL ES library. The engine proposed in this study was optimalized for executing 3D contents on a 2D-based systems and was designed to work in existing 3D supporting systems.

2. Realization of a 2D-Based 3D renderer

This study realized a 3D software renderer which can utilize 3D contents on a 2D based mobile system in order to overcome the problem of having a 3D mobile system to provide 3D contents in systems such as mobile phones and MP3 players.

Figure 1 shows the cores included in the mobile software renderer developed in the study. The cores were made based on OpenGL [19, 20], and provide a base for online and mobile data operation through support of a function of core sharing which is compatible with existing PC-based 3D resources. That is, they provide the environment where the PC-based 3D resource can be directly utilized for mobile contents without manual processing.

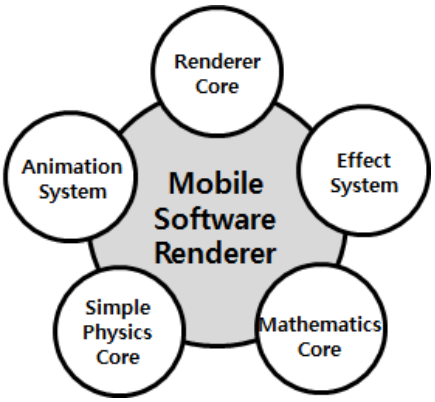

Figure 1. The Core of Mobile Software Renderer

Figure 2 shows the structure of the CM3D proposed as a mobile 3D renderer. It is composed of library-style software, which consists of in the following: The scene section which supports 3D composition, animation, and textures, the math section which supports math functions and geometric variation, the multimedia section which supports such functions as sound, image, and text, the network section which supports communication and data processing, the AI/SFX section which supports artificial intelligence and special effects, and the IO section which supports the functions for data processing and memory control.

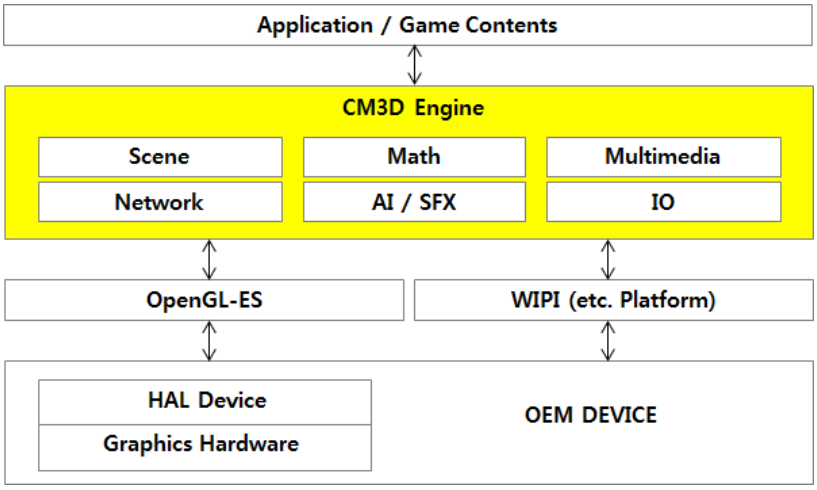

Figure 2. The Structure of Mobile 3D Renderer

Figure 3 shows composition of the developed renderer which is composed of library-type softwares. Each is composed of the scene section which supports 3D composition, animation, and textures, the math section which supports math functions and geometric variation, and the base section which supports the functions of sound, image, text and communication. The Scene section is composed of Mesh which supports 3D composition, Texture which supports surface processing, Camera which supports screen translate and rotation of screens, Bone which support the bone composition, Animation which supports animation-relevant functions, and Material which supports the texture of surface. The Math section is composed of Matrix, Vector, Scale, Rotation, Math, and Translate. The Base is composed of Sound, Image, Font, File, Memory, and Network. The library developed for this study is called from the mobile platforms to be realized to realize 3D contents and the 3D contents realized can be executed on a 2D mobile system.

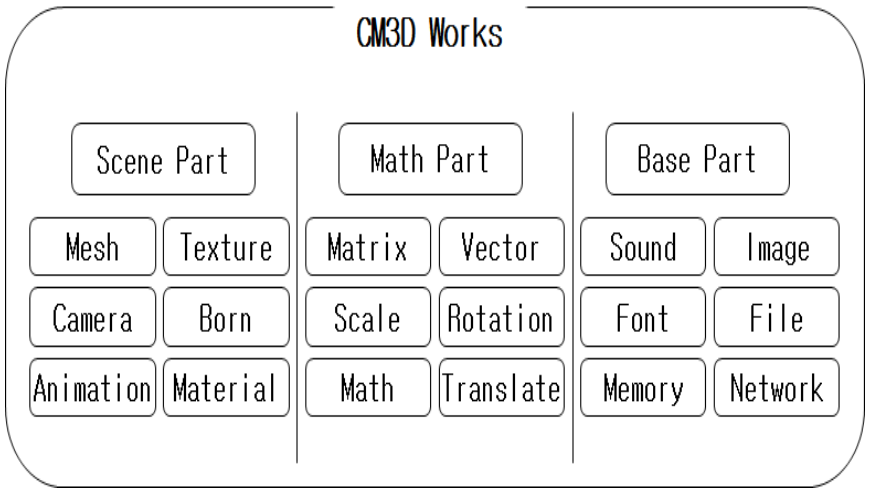

Figure 3. The Composition of Mobile 3D Renderer

Figure 4 presents a diagram of view flow on mesh objects of 3D renderer which supports execution of 3D contents on a 2D mobile system. It is composed of Mesh processing part, Texture processing part, and Mesh and Texture convergence part. For the Mesh part, if mesh file of 3D objects to be realized is loaded to the buffer, polygon list is stored in the vertex buffer and the polygons stored are indexed and sorted. UV list is stored in UV buffer and UVs are indexed and sorted.

For the Texture part, if the texture file of 3D objects to be realized is loaded to buffer, it is stored in image buffer and texture list is indexed and sorted. The Mesh and Texture convergence part works to connect texture lists which correspond to UV lists. A texture image is given to each polygon of 3D objects to be realized through projection matrix, view matrix, world operation and it is translated into 3D image. The same processing is given to all the polygons of 3D objects and after the processing is completed, each image is stored in screen buffer.

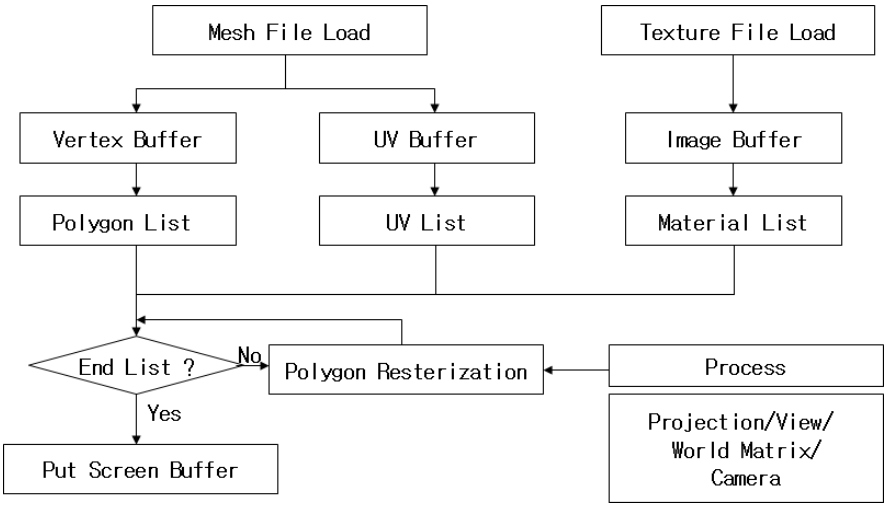

Figure 4. View Flow Diagram of Mesh object

Figure 5 is a diagram showing specific processing of Mesh Object. Model Data(Vertex, UV) and Texture Data are loaded to Buffer and the lists of each buffer are applied for Element List (Process Matrix Stack, Element List) and before storing pixel at render buffer for Element Resterization, the following processing is executed.

For ALPHA Color set, if pixel value of texture to be stored is the same as the value of Alpha Color, pixels are calculated instead of having the corresponding pixels taken. If not, intermediate value of corresponding pixel value at fixed render buffer and pixel value of texture to be taken is selected.

BLEND Pixel = (Render Buffer Pixel + Texture Pixel Color) / 2

For BLEND set, the processing is repeated until the last pixel and then it passes to next element After repeating the last Element List, data are stored at Screen Buffer.

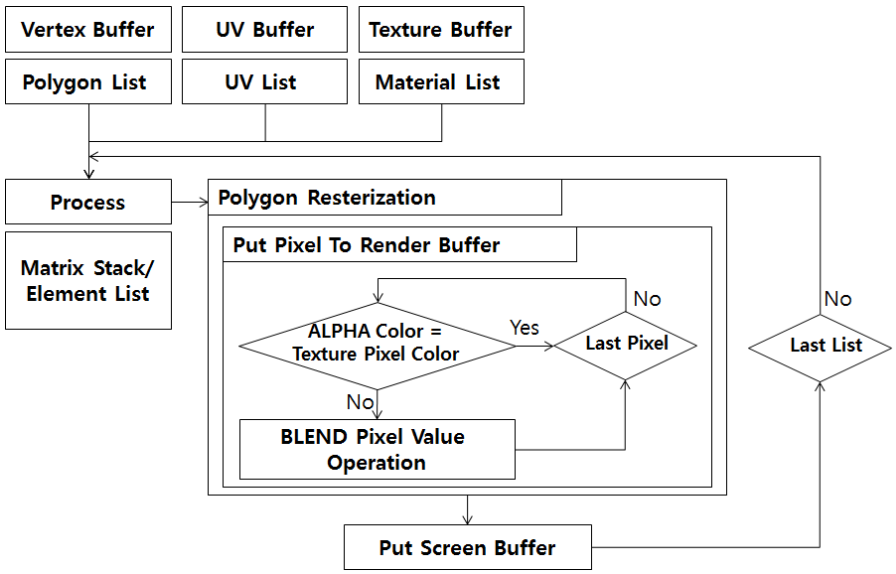

Figure 5. Specific Processing of Mesh Object

Figure 6 is a diagram showing processing of bone object for animation processing which is central at 3D renderer which supports execution of 3D contents on a 2D mobile system. It is composed of bone list and mesh list convergence part and animation part.

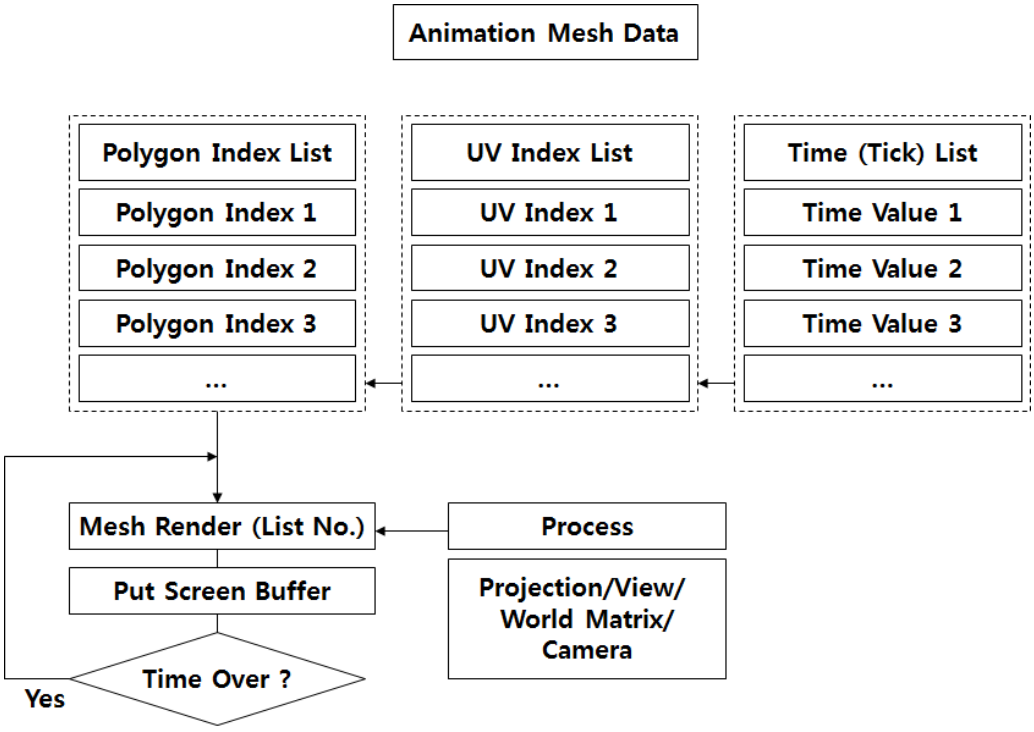

Figure 6. Animation Mesh Data Processing

The bone structure list is formed in stratum (body-arms, legs, face-hands, and legs), and each bone list has one to one corresponding relation with mesh list. Mesh 1 is rendered with coordinate value rotated as much as the rotated angle based on the rotational axis of Bone 1. For subsequent mesh, values of rotational axis and angle are operated based on mesh which is a mother node and rendering is executed. The behaviors are executed until the last mesh and if it ends, corresponding data are stored at screen buffer. If the fixed animation time elapses, the processing is repeated.

Figure 7 is a diagram showing animation mesh data processing in creation of animation which is central to 3D renderer. It consists of time list, UV list, and polygon list, and polygon and UV animation executes rendering by applying polygon and UV value changed according to animation time(Tick) of single mesh.

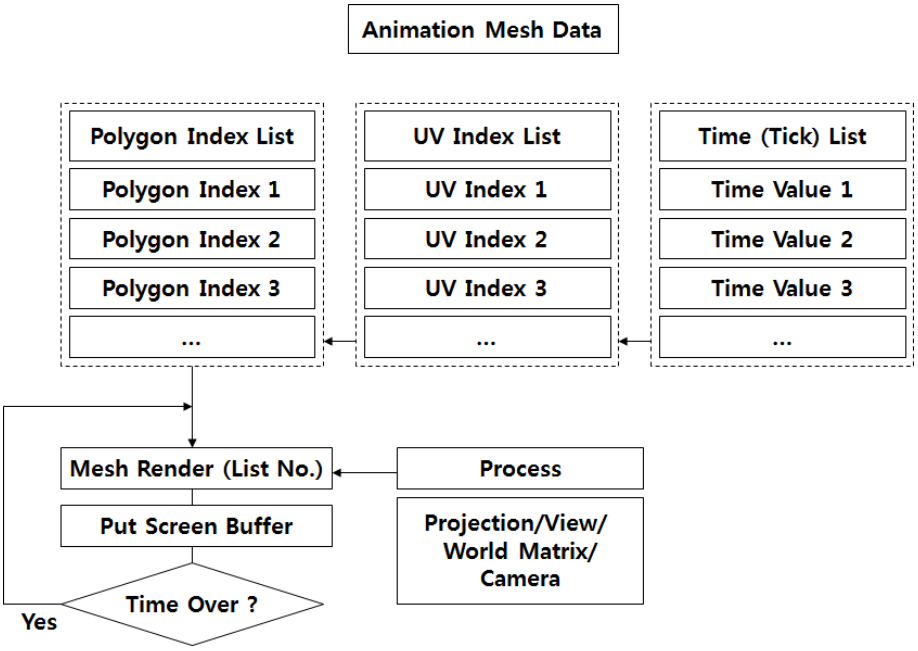

Figure 7. Animation Mesh Data Processing

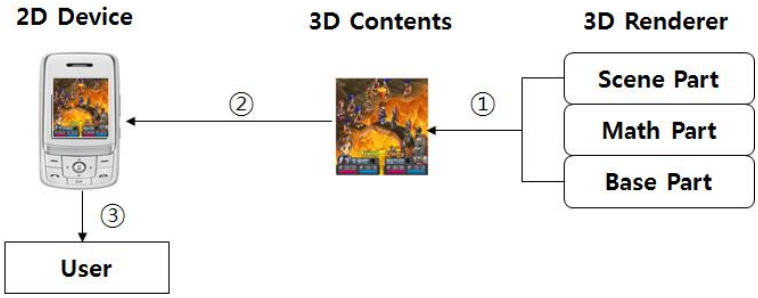

Figure 8. Application to Real Contents Production and Execution

Rendering is done for each list and the same process is performed until the end of the list If the process is completed, the corresponding value is stored in the screen buffer. Also, when the animation time elapses, the subsequent list mesh is repeated.

Figure 8 shows how 3D renderer developed in this study is applied for making and executing 3D contents. The process consists of making 3D contents using library, embedding 3D contents made at 2D/3D terminals and executing the corresponding contents at 2D/3D terminals.

Figure 9 shows the cross platform structure using the CM3D engine. The graphic resources made by a variety of toolkits are rendered according to the mobile and online platforms inside the engine and the resources which fit each system are provided for each system. Network and sound resources perform the same process and support operation on resource data and network resources. For this process, the mesh format file export engine and the true-type font file engine are used.

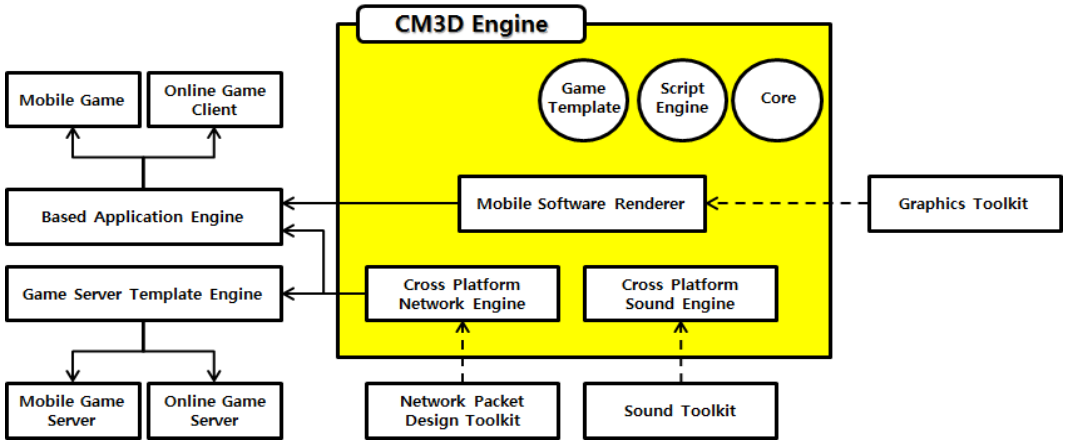

Figure 9. Structure of Cross Platform Using CM3D Engine

The developed 3D renderer has the following qualities and advantages in comparison with previous engine technologies:

First, it is a pure software rendering engine for easy programing work, for which the API is realized into the standard OpenGL. The realized engine is different from the existing engines and supports 3D contents in 2D systems as well as on 3D mobile systems.

Second, because of the extendibility of the application development, other engines support only full screen mode, but this application can adjust the size of the 3D area and location according to the use of the application as well as full screen mode. Also, through adjustment of the 3D areas, memory control and speed can be improved. As 2D image rendering and 3D model rendering are treated separately, various expressions and high speed can be achieved. In addition, this application provides extended math functions such as basic, matrix, vector, trigonometric function, and fixed point calculation functions which are needed for the development of a 3D program which is different from other engines.

Third, for better program control, this application has the following functions: treating real numbers with fixed point calculation, adjusting the precision of fixed points from 9 to 16 places, realizing the source using Register Variables and Macros, sorting into elements, ignoring hidden surface rendering, and the use of vertex animations for speed.

Fourth, model making methods are usually limited for easy work of designers, but this system shows contents in a view port without making changes in option settings in creating 3D models, and for 3D model file exports, this system provides contents in 3D MAX Script regardless of the 3D Max version while other engines support only specific versions.

Fifth, for various effects in rendering, 51 different element functions are provided so that the contents are realized in vertexes, lines, polygons, wire frames, and meshes according to the usages of 3D expressions. Also, in rendering, clockwise or counter-clockwise directions can be chosen, and in mapping, both blending of colors or texture, and the alpha test can be applied.

3. Simulation

To test performance of the renderer developed, this study simulated it using a PC-based emulator. The 3D resource data used for the simulation were character images made up of 500 polygons in 3D max. As a result of the simulation, in the test using the emulator, it was discovered that 60 frames were processed a second. Texture maps, color faces, texture wire-frames, color wire-frames, color vertices, blend texture maps, multiple models, and resizing models were also simulated.

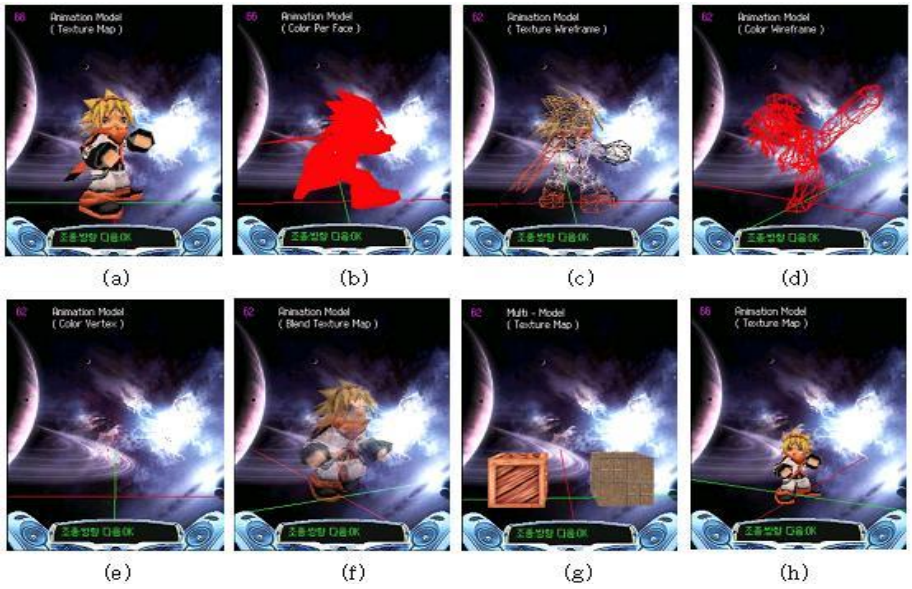

Figure 10. Tested Images on Emulator
(a) texture map (b) color face (c) texture wire-frame (d) color wire-frame
(e) color vertex (f) blend texture map (g) multi model (h) resizing model

Following this, the study tested the performance of the render in an actual 2D mobile phone. The SCH-B410 (Samsung Electronics) and SCH-V960 ((Samsung Electronics), as examples of 2D mobile phones, were used for the simulation and it was discovered that processing speeds were between 8 and 15 frames a second. Table 1 presents specifications of mobile phones used for the test.

Table 1. Specifications of Phones Used for Test

Spec.	SCH-B410	SCH-V960
memory	94MB	79KB
screen size	2.2inch	2.12inch
LCD	260K QVGA	262K Color
HD	240 x 320	240 x 320

Figure 11 presents the test images of a 3D character game on actual phones and test images made using the engine developed in the study on an emulator. (a) is a tested image on an actual phone, (b) is an example of a 2.5D based game developed in the study and (c) is an example of a game with the full 3D mechanism applied.

Figure 11. Engine Test Image (a) Character Test (b) 2.5D Game Test (c) 3D Game Test

Table 2 presents the results of the tests using the 500 polygons of data for comparison with existing mobile engines. The engine developed in the study was able to similarly execute 3D contents even in a system which supports only a 2D environment while existing renderers execute 3D contents only in systems which support 3D environment.

Table 2. Comparison with Existing Engines

Spec.	G3 Engine	M3D	HI Engine	Proposed Engine
frame/sec	20	7-10	20	8-15

4. Conclusion

This study developed a software renderer which makes 3D contents available on a 2D-based mobile system. The developed 3D renderer was composed of about 200 software libraries and each library was composed of a scene section, a math section, and a base. In this system, the 3D modelled data can be used without changes to the other settings, the environment, or the data, and the developed 3D contents can be used in 2D mobile systems the same way as existing 2D contents.

As a result of testing 3D character, 2.5D game, and 3D game using the library developed in this study on a 2D terminal, it was discovered that it has the similar performance in comparison with the results of the test using existing 3D engine on a 3D terminal.

These results provided a base on which existing 2D-based mobile contents are easily converted into 3D-based contents, and the function to access 3D contents on a 2D mobile

system using existing 3D data. Also, the CM3D engine developed in the study was designed to consider cross platform use, and through 3D resources, network resources, and an interface between sound resources, online and mobile resource can be shared and used. A fusion of independent content in platforms can be provided, and OSMU(One Source Multi Use) content can be developed.

The results of the study were applied to a PC-based emulator of a real mobile phone environment and similar results to existing 3D engines were attained. So, through this method the same content resources for both in PCs and mobile environments can be utilized.

In the future, all systems will change their environment so that they can support 3D, but as of now many mobile systems support only a 2D environment, the base for various 3D contents is given to 2D system users through the use of the renderer and the library developed, which will lead to activation of the 3D contents market.

References

[1] http://www.etnews.co.kr.
[2] http://kr.khronos.org/.
[3] http://www.jcp.org/en/jsr/detail?id=184.
[4] J. K. Cho, Y. H. Park and J. M. Kim, "Design and Implementation of Mobile 3D Engine using JSR-184 on J2ME", KIISE 2005 Fall Conference Proceeding, vol. 32, no. 2, **(2005)**, pp. 673-675.
[5] www.gomid.com.
[6] www.reakosys.com.
[7] www.hicorp.co.jp.
[8] B. H. Ko and S. G. Kim, "A Study on the Technology Tendency for Mobile 3D Game Engine", KOCON 2005 Fall Conference, vol. 3, no. 2, **(2005)**, pp. 19-24.
[9] D. K. Kim, J. M. Lee and C. H. Lee, "Design of Transformation Engine for Mobile 3D Graphics", The Journal of IEEK(SD), vol. 44, no. 10, **(2007)**, pp. 49-54.
[10] D. K. Kim, E. M, Kim and C. H. Lee, "Design of a Lighting Engine for Mobile 3D Graphics", IEEK Summer Conference Proceeding, vol. 31, no. l, **(2008)**, pp. 541-542.
[11] T. Zuvich, "Vehicle Dynamics for Racing Games", Game Developers Conference Proceeding, pp. 2000.
[12] H. C. Kim, D. K. Shin and D. I. Shin, "A Design of Context-Awareness Architecture and Game Physics Engine for Mobile 3D Racing Game", KIISE 2009 Korea Computer Conference Proceeding, vol. 35, no. 1, **(2009)**, pp. 46-50.
[13] Y. H. Song and K. S. Lee, "The Application and Development of 3D GUI Tool Frame for the Mobile Devices, KOCON 2009 Fall Conference, vol. 7, no. 1, **(2009)**, pp. 635-1258.
[14] C. H. Lee, "Design of Square Root and Inverse Square Root Arithmetic Units for Mobile 3D Graphic Processing", KIISE (SD), vol.46, no. 3, **(2009)**, pp. 1-118.
[15] H. Y. Lee and N. H. Baek, "OpenGL ES 1.1 Implementation Using OpenGL", The KIPS transactions. Part A, vol. a16, no. 3, **(2009)**, pp. 159-168.
[16] K. Y. Lee, "Design of a Variable-Length Instruction based on a OpenGL ES 2.0 API", Journal of IEEE Korea council, vol. 12, no. 2, **(2008)**, pp. 118-123.
[17] K. Pulli, T. Aarnio, K. Roimela and J. Vaarala, "Designing graphics programming interfaces for mobile devices", IEEE CG&A, vol. 25, no. 6, **(2005)**, pp. 66-75.
[18] C. H. Tu and B. Y. Chen, "The architecture of a J2ME-based OpenGL ES 3D Library", 9th Conf. on Computer Aided Design and Computer Graphics, **(2005)**, pp. 5.
[19] T. H. Kim and Y. B. Kim, "The mobile 3D three-dimensional studying system based in OpenGL ES", KMMS 2005 Spring Conference, **(2005)**, pp. 867-870.
[20] G. H. Hwang and S. H. Park, "Real-Time Rendering Toolkits for Mobile Games", KMMS 2007 Spring Conference, **(2007)**, pp. 1-831.