GPU 프로그래밍 기법을 이용한 비사실적 랜더링

Non-Photorealistic Rendering using GPU Programming Technique

블러르마 바트-오처*, 성경**, 김수균*

Bolormaa Bat-Ochir*, Kyung Sung**, and Soo-Kyun Kim*

요 약

컴퓨터 그래픽 기술 중 비사실적 랜더링 기술은 매년 발전을 거듭하고 있다. 비사실적 랜더링 기술은 일러스트, 애니메이션, 만화와 같은 예술적인 스타일에 영감을 받은 것입니다. 이러한 비사실적 랜더링 기술을 위한 많은 응용 프로그램들은 특히 애니메이션, 게임 산업 등에서 인기가 있다. 전통적인 컴퓨터 그래픽스에서는 비사실적 렌더링 기술에 많은 관심을 가지고 있지만, 많은 계산 시간을 요구하기 때문에 실시간으로 사용하지는않았다. 그러나 최근 몇 년 동안 비사실적 랜더링은 그래픽 가속기를 이용하여 훨씬 다양한 고급 랜더링 및 실시간 기술을 선보이고 있다. 본 논문은 비사실적 랜더링을 위한 GPU 프로그래밍 기법에 대해 설명한다.

Abstract

NPR(Non-Photorealistic rendering) technique is developing by every years. NPR is inspired on artistic styles, which is painting, drawing, technical illustration, animation and cartoon. There have many application programs for NPR, which is popular and useful of animations, even on game industrial. In traditional computer graphics focused on non-photorealism, but this method need much more memory and time. Recent years, Many NPR methods present advanced rendering technique and real time technique using graphic accelerator. This paper propose to explain NPR with GPU programming.

Key words : GLSL, HLSL, 쉐이딩, GPU 프로그래밍, 비실사적 랜더링

Ⅰ. Introduction

Computer graphic processing and normal processing work with CPU(Central Processing Unit) in earlier years. The 3D interactive graphics expended much more time for rendering, when the used only CPU. If model is high level shading, then it has to render much more time and low level graphics match in 3D rendering. Lately advanced graphic systems have been use the GPU(Graphics Processing Units)[1],[2].

GPU programming is API(Application Programmer's Interface). This is the side of the application programming, it is widely using especially game [3] and animations. When the start to use GPU, it gives chance

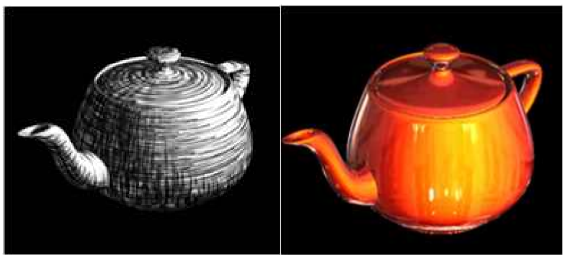
* 배재대학교 게임공학과(Department of Game Engineering, PaiChai University)
** 목원대학교 컴퓨터교육과(Department of Computer Education, MokWon University)
· 제1저자 (First Author) : 블러르마 바트-오처, · 교신저자 (Correspondent Author) : 김수균
· 투고일자 : 2011년 11월 26일
· 심사(수정)일자 : 2011년 11월 26일 (수정일자 : 2011년 12월 21일)
· 게재일자 : 2011년 12월 30일

to get have good result and save the time for rendering. Recently some works are aimed to rendering on smart phone [4].

In this paper focused on NPR style, especially cartoon shading (cel shading). There is few GPU programs are used for graphics. Here is explaining few of GPU programs. We present differences of some GPU programs and compare to this results.

Ⅱ. Related Work

First, it is need to explain cartoon shading. Especially, NPR is the contrast of the photorealism. Recently, NPR is important and impressive branch of computer graphics. The model in 3D environment, NPR works to increased availability of programmable GPU's and applies to the rasterised image, then result is displayed on the screen. NPR techniques attempt to create images or virtual worlds visually comparable to renderings produced by a human artist [5].



(a) Hatching (b) Two-Tone Car Paint
그림 1. 렌더멍키를 이용한 비사실적 렌더링효과
Fig 1. NPR effect using Rendermonkey

The NPR technique is based in the earlier years, for enhanced visual comprehensibility with 3D images. This rendering process used Geometric Buffers (G-buffers) at that time[6].

Now, the several artwork styles have been explored in the NPR literature, such as pen and ink[7], painting[8], engraving, informal sketching, charcoal drawing and watercolor in [9]. NPR algorithms mostly focus on a specific artistic style. In animation techniques used impressionistic painting and cartoon style rendering[8].

Figure 1 shows 3D teapot model using rendermonkey with texture effects by hatching and two car paint.

Ⅲ. Programming language and shader

There are using with few programming languages in now days. Specially, used with game programming and other reason. Each one of that based on C/C++ and program will be use graphic pipeline. Our purpose is explaining those languages and focus with each of that, and explain to what's the best thing or differences. Our goal is the making NPR for use with different kinds of programming languages. Especially, the goal of result is cartoon shading (cel shading).

3-1 GLSL

OpenGL, is the popular computer graphic's application, using with cross-platform languages. This application produces 2D and 3D computer graphics and developing from Silicon Graphics. First version OpenGL 1.0 is released on 1992, latest version OpenGL 4.1 released on 2010. The OpenGL has single library named GL. Shaders are written by GLSL(OpenGL Shading Language), which is separate from OpenGL[1],[2]. Our purpose use GLSL and greating cartoon shading in OpenGL.

The OpenGL as a pipeline, with processing stations along the way. That takes in the vertices of 3D object space and transforms them into screen coordinates[1]. The 3D coordinates of vertices enter on the left, and undergo a series of transformations and operations include:

1. Clipping away vertices.
2. Determining visible surface and which are behind the other surfaces.
3. Rasterizing polygons and drawing lines.
4. Shading and texturing of the pixels making up the polygonal faces.

5. Performing a number of image processing operations such as bluring and compositing.

Shading process has working with vertex and fragment shader.

There are 2 ways of doing cel-shading. Multiple texturing and some case use the sharp lighting map. Multi-texturing way is not every graphics card supports it. Switching the roles of the texture and vertex color. Instead of the texture shading the color, the color is going to shade the texture.

3-2 HLSL

The primary theme of HLSL(High Level Shading Language) is vertex and pixel shaders, which replace sections of the fixed function pipeline with a custom program that we implement, called a shader. Shaders are completely programmable and allow us to implement techniques that are not defined in the fixed function pipeline. Consequently, the number of techniques that we have available at our disposal has greatly increased. The programmable sections of the rendering pipeline are commonly referred to as the programmable pipeline (see figure 2).

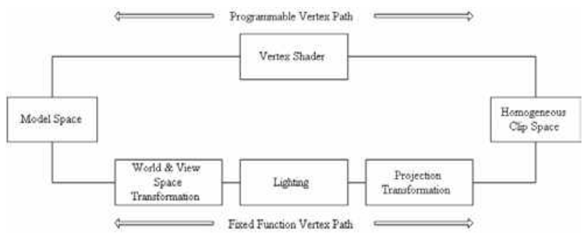

Programmable Vertex Path
Vertex Shader
Model Space
Homogeneous Clip Space
World & View Space Transformation
Lighting
Projection Transformation
Fixed Function Vertex Path

그림 2. HLSL 파이프라인
Fig. 2. The vertex shader replaces the lighting and transformation stage of fixed function pipeline

DirectX is one of the application programming interface, developing from Mficrosoft. Important on game development. There is released to Directx 1.0 to DirectX 11.0 version[11]. 3D computer graphics using DirectX 9.0, and an emphasis on game development[10]. DirectX is a set of APIs for developing multimedia applications on the Windows platform. Also named with Direct3D. DirectX is programming shaders to use HLSL(High Level Shading Language). There is a cel shading for HLSL. It's also similar with OpenGL shading.

3-3 RenderMonkey

RenderMonkey has simplest user interface, the quick development of shaders and effects. Shaders are more than just code, these require a framework that takes care of setting up other needed components such as geometry and textures[13].

Also shader development tool should help bridge the gap between artists and programmers by removing any API dependencies. Such a tool should also remove the need for programmer intervention in the development of simple or complex shaders. The framework for the tool must be flexible enough to allow it to adapt to future technology needs.

RenderMonkey developed by ATi. With its component-based architecture. It makes shader development a simple task without making you spend much time setting up your scene. RenderMonkey working with:

- RenderMan compiler
- DirectX8 low level assembly support
- DirectX9 High Level Shading Language (HLSL) support
- DirectX11 High Level Shading
- OpenGL (GLSL)

3-4 CG

Cg is language and system, called C for graphics. It is developed by NVIDIA[12]. It is based on C languages. Latest version came Febuary 2011 version of Cg 3.0. Cg is both a language and system. Made of goal was easier programming of GPU and it does not replace a general programming language. Cg compiles to code that can be executed by a GPU's programmable fragment or vertex processor[8]. And also cross-platform closely working for OpenGL and DirectD3D. Working with OpenGL ARB (Architectureal Review Board) as well as than Direct3D. CG compiler produces OpenGL or Direct3D code, drivers perform final translation into hardware-executable code.

3-5 CUDA

CUDA is a Compute Unified Device Architecture, developed by Nvidia. This is the parallel computing, engine in Nvidia graphics processing units (GPUs), released in 2006. Programmers use 'C for CUDA' [9]. CUDA gives developers access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs. In computer game industrial's GPUs use with game physics calculations. Also CUDA provides low and higher level API.

CUDA is an attractive platform

- Broad - OpenGL, DirectX, WinXP, Vista, Linux, MacOS- Widespread
- Over 85M CUDA GPUs, 60K CUDA developers.

Ⅳ. Compare of programming languages

1. OpenGL defines a state machine controls, that the rendering pipline. Vertices defined by graphic primitives to the pipeline. OpenGL program is not popular in computer graphics. The code is not for beginners, it is basically top graphics. But results are almost no differences from other programming results.

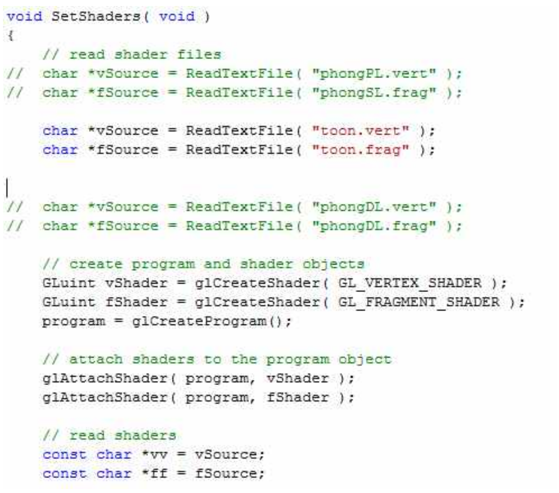
2. DirectX is similar with OpenGL. Basically DirectX mostly use in game industrial. This shading is created by microsoft, so program will working not too problems. It is one of the top level programming languages.

3. Cg is popular nowadays. It has easy programming progress and also works with OpenGL and DirectX. Result is good and not taking time. But, not working with all computer graphic cards. It is middle level of programming

4. Cuda - It is also made from Nvidia. Result is a good, one of the newest program. But is already became popular in these days.

5. RenderMonkey is basically easiest. Programming interface is easy and codes are not too complicated. It is low level of programming languages.

There is have some result of different programming resource code. Even it is rendering similar results, hardware and computing time, processes are different from each others.



```
void SetShaders( void )
{
    // read shader files
//  char *vSource = ReadTextFile( "phongPL.vert" );
//  char *fSource = ReadTextFile( "phongSL.frag" );

    char *vSource = ReadTextFile( "toon.vert" );
    char *fSource = ReadTextFile( "toon.frag" );

|
//  char *vSource = ReadTextFile( "phongDL.vert" );
//  char *fSource = ReadTextFile( "phongDL.frag" );

    // create program and shader objects
    GLuint vShader = glCreateShader( GL_VERTEX_SHADER );
    GLuint fShader = glCreateShader( GL_FRAGMENT_SHADER );
    program = glCreateProgram();

    // attach shaders to the program object
    glAttachShader( program, vShader );
    glAttachShader( program, fShader );

    // read shaders
    const char *vv = vSource;
    const char *ff = fSource;
```

그림 3. 예제: 툰 쉐이딩 [5]
Fig. 3. Example: Toon shading [5]

Figure 3 present example code interactive computer graphics [5]. It uses with GLSL source coding in OpenGL. Figure 4 shows result of this example code, also present cel shading with silhouette.

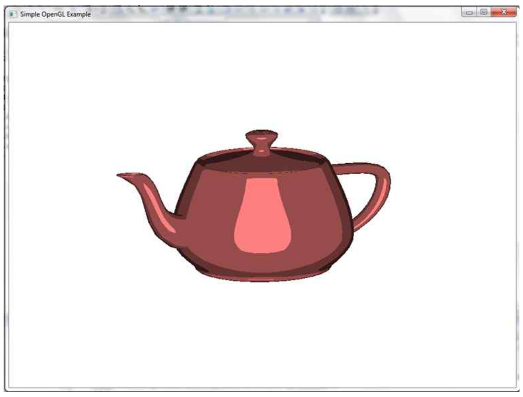
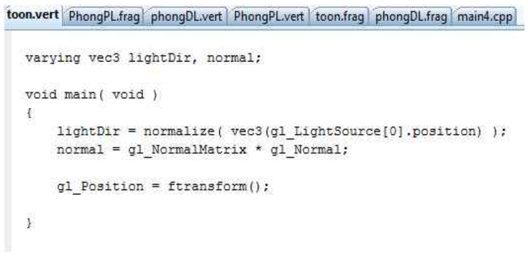


그림 4. GLSL 결과
Fig. 4. Result of GLSL

Also, the code(See figure 5) is part of Figure 4.



```
varying vec3 lightDir, normal;

void main( void )
{
    lightDir = normalize( vec3(gl_LightSource[0].position) );
    normal = gl_NormalMatrix * gl_Normal;

    gl_Position = ftransform();

}
```

그림 5. 예제: 툰 쉐이딩 [5]
Fig. 5. Example: Toon shading [5]

Here is the shading coding with toon shading.

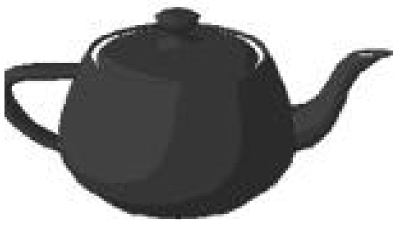


그림 6. DirectX를 이용한 셀 쉐이딩 [9]
Fig. 6. Cel shading using DirectX 9.0 [9]

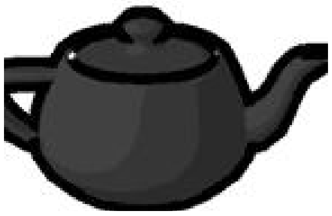


그림 7. DirectX를 이용한 스토크를 가진 셀 세이딩 [9]
Fig. 7. Cel shading with stroke using DirectX 9.0 [9]

Figure 6 and 7 show the HLSL coding result in DirectX. This result is created by cartoon shading. Figure 7 present cel shading with stroke.

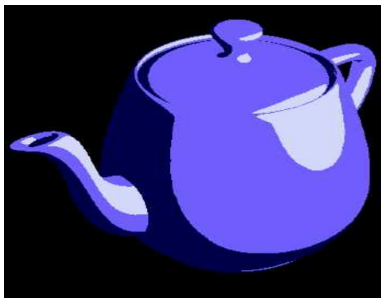
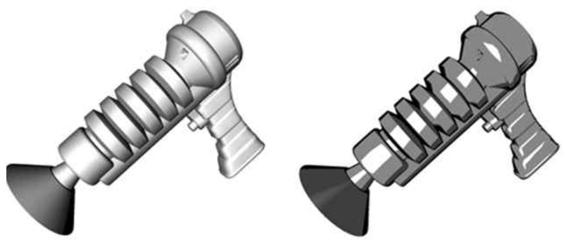


그림 8. 스토크를 가진 셀 세이딩 [9]
Fig. 8. Cel shading with stroke using Rendermonkey[9]



(a) without texture (b) with texture
그림 9. 툰 세이딩 [10]
Fig. 9. Toon Shading(The Cg Tutorial 2003[10])

Figure 8 shows result of cel shading using RenderMonkey. And also showing with stroke line. Figure 9 shows result of toon shading using Cg technique, Figure 9b does not include texture information.

Ⅴ. Conclusion

This paper present to explain NPR with graphic accelerator. The 3D interactive graphics reduce time for rendering using GPU. It is widely using especially game engine and animations. When the start to use GPU, it's giving chance to get have good result and save the time for rendering. We describe different GPU methods for NPR techniques.

References

[1] Edward Angel and Dave Shreiner, “Interactive Computer Graphics 6th edition” 2011.

[2] Edward Angel, Interactive Computer Graphics 5th edition, 2008.

[3] 안성옥, “새로운 숫자 정렬화 큐브 퍼즐 놀이인 큐스타 게임 ”, *한국지식정보기술학회 논문지,* 제5권 2호, pp. 133-139, 2010

[4] 김선정, “iPhone용 3차원 점 집합 가시화 ”, *한국지식정보기술학회 논문지,* 제5권 6호, pp. 262-268, 2010

[5] Bruce Gooch, “Non-Photorealistic Rendering” 2001.

[6] Takafumi Saito and Tokiichiro Takahashi, “Comprehensible Rendering of 3-D Shapes”, In *Computer Graphics*

(SIGGRAPH '90 Proceedings), volume24, pages 197-206, August 1990.

[7] Georges Winkenbach, David H. Salesin, "Computer-Generated Pen-and-Ink Illustration", In Proceedings of *SIGGRAPH '94 (Orlando, Florida, July* 24-29, 1994), Computer Graphics Proceedings, *Annual Conference Series, pages 91-100. ACM SIGGRAPH, ACM Press*, July 1994.

[8] Barbara J.Meier, "Painterly Rendering for Animation ", In Proceeding SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[9] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H., "Computer Generated water color", 1997.

[10] Frank D. Luna, "Introduction to 3D Game Programming with DirectX 9.0" 2003.

[11] Frank D. Luna, "Introduction to 3D Game Programming with DirectX 10", 2008.

[12] Craig Peeper Jason L. Mitchell, "Introduction to the DirectX 9 High Level Shading Language"

[13] Thomson Course, "Shaders for Game Programmers and Artists", 2004.

[14] Randima Fernando, Mark J. Kilgard, "The Cg Tutorial", *NVIDIA Corporation,* 2003.

[15] Jason Sanders, Edward Kandrot., "Cuda by Example An Introduction to General-Purpose GPU Programming", 2010.

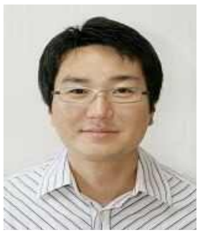
블러르마 바트-오처

배재대학교 게임공학과 재학중

성 경 (成 鏡)

2003년 한남대학교 컴퓨터공학과 (공학박사)
1994년~2004년 동해대학교 컴퓨터공학과 교수
2004년~현재 목원대학교 컴퓨터교육과 교수
관심분야 : 정보보호 및 정보관리, 컴퓨터네트워크, 신경회로망, 컴퓨터교육

김 수 균 (金修均)



2002년 2월 : 고려대학교 컴퓨터학과 (이학석사)
2006년 2월 : 고려대학교 컴퓨터학과 (이학박사)
2006년 3월~2008년 2월 : 삼성전자 통신연구소 책임연구원
2008년 3월~현재 : 배재대학교 게임공학과 전임강사
관심분야 : 컴퓨터 그래픽스, 모바일 그래픽 등