Non-Photorealistic Rendering using GPU Programming Technique

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

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, Shading, GPU Programming, Non-Photorealistic Rendering

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
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].

![Fig 1. NPR effect using Rendermonkey](image)

(a) Hatching  (b) Two-Tone Car Paint

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.