MAT 594G / CS 280: Advanced Computer Graphics	Oct. 25, 2012
Fall Quarter, 2012	Assignment 2

Due by **midnight, the evening of November 6, 2012** via GauchoSpace.

1 GLSL Shading

In this assignment, you will get experience writing basic GLSL shaders, and experiment with the notion of deferred shading.

1.1 Basic GLSL Shading

As a basic first shader, we will be computing *Julia sets* (Figure 1). Julia sets are a type of fractal that are fairly simple to compute, and map extremely well to graphics hardware. They are closely related to the more famous *Mandelbrot set*.

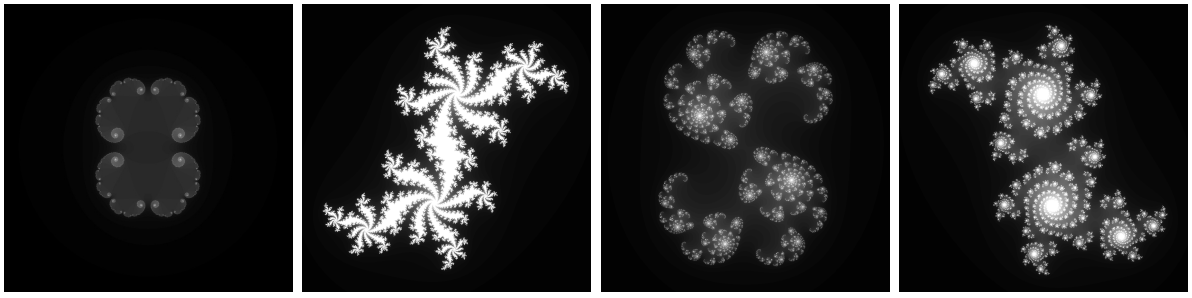

Figure 1: Julia set fractals. For reference, the leftmost image is viewed from $x = (-2.5, 2.5)$, $y = (-2.5, 2.5)$, and is set to $c_{real} = 0.285$, $c_{imaginary} = 0.0$.

The OpenGL Shading Language (a.k.a. The Orange Book) describes Mandelbrot and Julia sets in Chapter 18. The starter code contains a project *julia* that sets up a basic textured quad. In addition to getting basic Julia set computation off the ground, add the following features:

- Store the Julia set constant $\boldsymbol{c}$ as a `uniform` in your shader, and allow it to be dynamically modified at runtime using the arrow keys. You will need to call `glGetUniformLocation` and other related functions, and modify the body of `glutSpecial`. In your final program, you should be able to explore the space of Julia sets in real-time.
- The simplest coloring scheme, used in this document's images, is to simply divide the number of computed iterations by the maximum allowable iterations. Devise something more interesting.

1.2 Deferred Cartoon Shading

As an introduction to deferred shading, we will look at some cartoon-style rendering effects. There are many methods of performing cartoon, or cel, shading, but we will implement a fairly straightforward one here. The starter project *deferredViewer* implements a deferred diffuse renderer for you. It currently outputs a diffusely shaded bunny, and also stores the depth, normal and transformed light position in a variety of available textures (Figure 2).

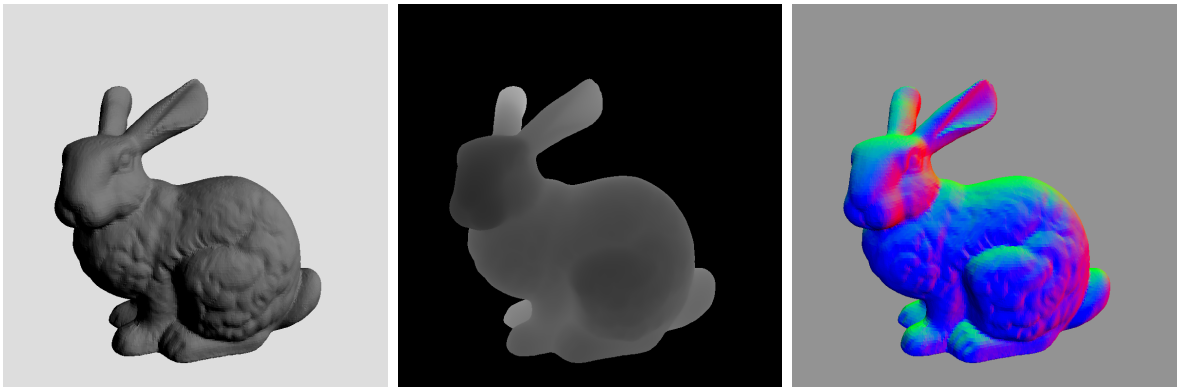

Figure 2: Left to right: Diffusely shaded bunny (starter code default), the depth texture, and the normal texture.

1.2.1 Ink Outlines

The first key element to a cel shading look is the appearance of heavy strokes along the profile of the model:

In order to achieve this look, for each shaded fragment, compare its current depth to the depth a small distance away in the positive and negative x and y directions in the depth texture. If the difference is large, we must be looking at an edge, so the fragment should be shaded black.

1.2.2 Cartoon Lighting

The second key element of cel shading is a non-smooth shading model. The simplest way to achieve this is to clamp the diffuse and specular colors based on some threshold, e.g. if the diffuse component is less than or equal to 0.5, clamp it to zero, otherwise set it to 0.5 (Figure 3, left).

Figure 3: Left to right: Diffusely toon-shaded bunny, with diffuse, specular, and outlines (zoom in to see blocky artifacts), and the smoothed model.

Note however that the starter code *only contains diffuse lighting.* You will need to implement your own deferred specular Phong lighting first before you can threshold it. You may notice that when your cartoon shader is working, the relatively coarse normals of the bunny model become more apparent (Figure 3, middle). One approach to reducing this artifact is to smooth the model itself (Figure 3, right). However, there is more than one way to address this problem. Devise and implement an approach that addresses this artifact.

2 Points Breakdown

Turn in your files via GauchoSpace in a ZIP file named `HW2-<UCSB Net ID>.zip`. Include a `README.txt` that describes what platform you developed on, how you dealt with the normal artifacts, and any special instructions for building and running your code you think may be necessary.

1. Get basic Julia sets working in GLSL. **(20 points)**
2. Get keyboard interaction working with Julia sets. **(10 points)**
3. Implement a novel Julia set coloring scheme. **(10 points)**
4. Get deferred ink outlines working in cartoon shading. **(20 points)**
5. Get deferred specular shading working in GLSL. **(15 points)**
6. Get deferred cartoon diffuse and specular shading working. **(15 points)**
7. Address normal artifacts. **(10 points)**

For each of the items above, with the exception of #2, submit a numbered screenshot that shows the feature working. E.g. `4.jpg` should be a screenshot of the bunny with just an ink outline.

3 Compiling the code

The starter code comes with two `Makefiles`, `Makefile.mac` and `Makefile.linux`. If you are running on a Mac, typing `make -f Makefile.mac` from one of the starter code directories should build the code. If you are running on Linux, you should be able to build using `make -f Makefile.linux` after making the following changes.

In all of the header and source files(.h and .cpp) replace:

```
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
```

with:

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
```

In VEC3F.h add:

```
#include <cstdio>
```

For the `deferredViewer` project, also add the following line in `OBJ.cpp`:

```
#include <cstring>
```

Thanks to Sahar for figuring all this out. We also tried building in Windows under Cygwin, but were not successful. If you are able to build successfully using either Cygwin, MinGW, or Visual Studio, please let us know.