Winter 2016, CS 112
Programming Assignment 4
Regular Submission Due: March 7th, 2016, 11:59 PM PST

PROJECT GOAL: Implement your own vertex and fragment shader. Implement geometric transformation, lighting, and texture in OpenGL.

1 Geometric Transformation

In OpenGL 3.0 and up, OpenGL expects the developer to maintain their own model view and projection matrices. In this portion of the assignment, take your matrix transformation code from programming assignment 2 and port the code over to programming assignment 4's solution. You will need to apply the appropriate transformation to the viewMatrix and the projMatrix and pass those matrices to the vertex shader using uniform variables as provided.

In the vertex shader (vert.txt), you will find that we have provided you with:

```
uniform mat4 viewMatrix, projMatrix;

in vec4 position;
in vec3 color;
in vec4 normal;

out vec3 Color;
out vec4 Normal;

void main()
{
	gl_Position = ? * position ;
}
```

viewMatrix and projMatrix are the uniforms that gets passed in from your C++ code. Position, color, and normal are attributes.

The main function gets called on every single vertices in the scene.

You need to fill in the code to produce a transformed vertex given its position.

For example:
gl_Position = Matrix * position; //Where Matrix is your transformation matrices.

2 Lightning: Gouraud and Phong shading

Part 1: Gouraud shading

Implement the light equation and compute the color at every vertex in the vertex shader. You will be moving your illumination equation (from assignment 3) into the vertex shader.

Part 2: Phong shading

To implement Phong shading, you will need to use the fragment shader to calculate the color for each pixel. Fragment shader gets called on every pixel. OpenGL will interpolate normals, so you have the normal vector for each pixel.

```
#version 330
uniform vec4 Ambient;
uniform vec3 LightColor;
uniform vec3 LightPosition;
uniform float Shininess;
uniform float Strength;

uniform vec3 EyeDirection;

in vec3 Color;
in vec3 Normal;
in vec4 Position;

out vec4 outputF;

void main()
{
	outputF = ?;
}
```

3 Texture Mapping

Texture mapping is the process of mapping images onto the faces of objects. Just like in vertex coloring, where you assign a color to a vertex, you will need to calculate and assign texture coordinates to the vertex. OpenGL will use these texture coordinates to map sample the image and map it onto a face. Since texture mapping deals with the actual pixel color value, you will be using the fragment shader to implement texture mapping.

4 How to Do It

Download the PA4 software package from the EEE dropbox. Open the solution using Microsoft Visual Studio. The solution was created inVS2012, but you should be able to upgrade it to VS2013. All the necessary libraries have been included in the project.

As you may notice, the structure of this package is a little different from the previous three packages. The entire program is contained in main.cpp with a couple of sample shaders for you to start with – this should simplify the submission process. The project is divided up into three parts: geometric transformation, lighting, and texture mapping.

Geometric transformation is worth 20%, lighting is worth 40%, and texture mapping is worth 40% of the project grade.

To receive full points for the geometric transformation part of your project. You need to complete the vertex shader and save it as vertgeometry.txt. We have provided you with all the necessary parameters. All you need to do is calculate the new vertex position given the model view and the projector matrices and store the result in gl_Position.

To receive full points for the lighting part of your project you will need to create two sets of shaders.

For Gouraud shading, you first need to complete the vertex shader and save it as vertgouraud.txt. You will need to include the transformed vertex code from vertgeometry.txt in this shader. You will also need to calculate the vertex color and store the result in attribute Color. You will use the same illumination equation from PA3 to calculate the color value for a given vertex. You will use the fragment shader frag.txt for Gouraud shading. Save a copy of the fragment shader as fraggouraud.txt.

For Phong shading, you need to complete the vertex shader and save it as vertphong.txt. You will need to include the transformed vertex code from vertgeometry.txt in this shader.

Project Presentation
Please put all of your project files in a folder. Then compress this folder (.zip or .rar) and upload it to EEE Dropbox before the deadline. Then you should come to one of the following lab sessions to run the code and show your work.

Tuesday 3/8/2016 9am-9:50am location: ICS364B
Wednesday 3/9/2016 8pm-8:50pm location: ICS364B
Thursday 3/10/2016 9am-9:50am location: ICS364B
Monday 3/14/2016 8pm-8:50pm location: ICS364B