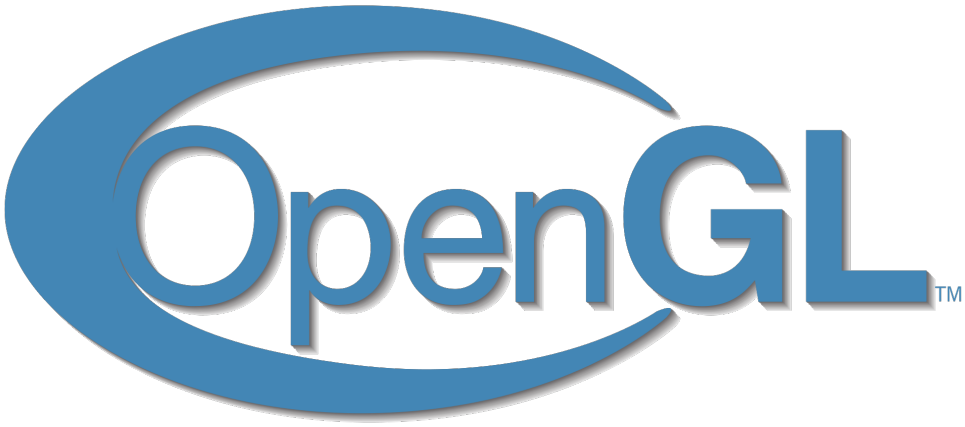
OpenGL Programming Assignment
Due: 16 Dec, 2020, 6:00pm



In this assignment you will learn OpenGl and shader programming. You will implement your own vertex and fragment shader.

We provide a zip file with one demonstration program. The provided code is created by Microsoft Visual Studio and its in C++ language. But you can use any other IDE for development of this assignment.

Part 1: Geometric Transformation (40 points)
(8 days)

In OpenGL 3.0 and up, OpenGL expects the developer to maintain their own model view and projection matrices. 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.

```
uniform mat4 viewMatrix, projMatrix;

in vec4 position;
in vec3 color;
in vec4 normal;
out vec3 Color;
out vec4 Normal;

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

In the vertex shader (vert.txt), you will find that we have provided you with:
Step1: 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.

Step2: In this step you need to write the code to create different transformation matrices. Open the main.cpp file and complete following functions:

```
setScale()          // This function returns scale matrix
setTransMatrix()    // This function return translation matrix
rotationMatrix()    // This function returns rotation matrix.
setIdentMatrix()    // This function return identity matrix
```

After finishing this part you should be able to run the code and see a blue cube.

Part 2: Lighting (40 points)
(8 days)

Step1: Gouraud shading (20 points)

Implement the light equation and compute the color at every vertex in the vertex shader. For this purpose you need to calculate the lighting for each vertex using following equation:

Vertex color = (material emission) + (global ambient)*(material ambient) +
[(light ambient)*(material ambient) + (max{L.N, 0})*(light diffuse)*(material diffuse) + (max{H.N,0})n *(light specular)*(material specular)]

H=(L+V)/|L+V|, where L is the unit vector towards the light source, and V is the unit vector towards the eye. This is called the half-vector between L and V.
N is the normal vector. n defines the shininess. The “max” functions will make sure that the light is not shining in the back-side of the face. If this value goes beyond 1, it is clamped to 1 (lighting has been saturated at that point).

Step2: Phong shading (20 points)

For implementing the phong shading you need to calculate the lighting for each pixel. For this purpose you need to use the fragment shader which gets called for every pixel.

Part 3: Animation (20 points)
(4 days)

In this part you are going to create an simple animation. You should create two cubes. Let us call these two cubes Spinner and Revolver, and they should satisfy the following constrains.

1. Spinner should rotate about an axis through its center, parallel to any edge.
2. Revolver rotates about the same axis as Spinner revolving around Spinner.
3. Revolver DOES NOT rotate about an axis through its own center.
4. The frequency of Spinner rotating about its own axis should be the same as that of the rotating Revolver. This means, for every complete revolution of Revolver around Spinner, Spinner should itself have completed 360 degrees.
5. Spinner should be twice the size of Revolver.

Deliverable

You should create three different versions of vertex and fragment shader. The first one should show the cubes with out lighting. The second one should show the cubes with Gouraud shading and the last one is for Phong shading. The deliverable is a short presentation of the demo (described below) made via zoom screen sharing on 16 Dec during class time.

During the demo you should run your code with the animation. Then you should be able to switch between different lighting methods by pressing following keys on the keyboard.

- Pressing “a” should show the cubes without any lighting.
- Pressing “p” should show the cubes with Phone shading.
- Pressing “g” should show the cubes with Gouraud shading.