COSC 4370 - Homework 3

Rizwan Lokhandwala PSID: 1887820

October 2022

1 Problem

The assignment was to display a cube by implementing a 3D viewing and Phong shading model.
main.cpp, camera.h, phong.vs, and phong.frag were to be modified.

2 Method: Displaying the cube

The first goal was to display the cube on screen. The vertices for the cube were already provided in the format as follows:

vertices[] = x,y,z,xNormal, yNormal, zNormal,

where the first 3 represents the vertex and the second three represent the normal vector to it's surface.

vertices[] held the information for all 6 faces and had 6 vertices for each face.

This is because OpenGL mainly works best with triangles. So for each face of the cube, two triangles were drawn. To elaborate, if you had a square ABCD, OpenGL would draw triangle ABC and triangle CDA. Meaning two coordinates are shared for each face.

The triangles were already mapped out and just needed to be projected onto the screen. In order to do this, I wrote a simple perspective projection matrix using glm::perspective(). (in main.cpp)

In camera.h, getViewModel also needed to be completed. To do this, I created a 4x4 identity matrix and named it viewMatrix. This was done using the method: glm::mat4 viewMatrix = glm::mat4(1.0f);

I then used the method:

viewMatrix = glm::lookAt(Position, Position + Front, Up);

which sets the camera position to the vector (Position, Position+Front, Up)

Finally to display the cube, I had to set put it all together to project the cube vertices to the screen. This was done using the formula:

screenPosition = projectionMatrix * viewMatrix * modelMatrix * vec4(position, 1.0);

With this the cube was displayed on screen, but with no lighting you could not tell where the edges of the cube are.

3 Method: Lighting and Shading

Next came the lighting/shading of the cube. This is done in 3 steps, Ambient Lighting, Diffusion, and Specular Lighting. First we will discuss Ambient Lighting.

4 Implementation: Ambient Lighting

First I set the ambienceStrength = 0.1; and created a vec3 ambientLighting = ambientStrength * lightColor;

This means that no matter where the light source is coming from, some constant amount of light will be reflected off the cube.

5 Implementation: Diffusion

In order to know how much lighting gets diffused, we need to know the direction that the light is facing, and the normal vector to the surface. First we determine the unit vectors for the normal and

direction of the light. We then take the dot product of the two unit vectors to know the strength of the diffusion. (use of max() makes certain that diffusion strength ¿= 0)

From there we set diffuseLighting = diffuseStrength * lightColor.

6 Implementation: Specular Lighting

Specular lighting represents how a direct light source might behave when reflected off a shiny surface. In order to accomplish this, we initially set a specular strength of 0.5. This ensures that the location closest to the light will stand out, but not take too much attention away from the rest of the cube.

From there, we determine how much of the reflection of the light bounces towards the view. We do this by taking the dot product of the view direction and the reflection direction. We then set the value of specularLighting = specularStrength * spec * lightColor;

7 Implementation: Final Lighting

Finally, we combine all the lighting to determine the color of each position.

vec3 Color3 = (ambientLighing + diffuseLighting + specularLighting) * objectColor;

But the color needs to be a 4 dimensional vector, so we set color = vec4(color3, 1.0);

8 Results

My results ended up looking very similar to the desired image. The color is the same, the faces have the proper shading, and I attepmted to match the specular lighting/ view angle to the best of my ability.