COSC 4370 – Homework 3

Name: Caitlin Dooley PSID: 1389102

October 2021

1 Problem

The goal of this assignment was to recreate the an image of an coral colored cube by implementing the Phong shader model using OpenGL. References used to complete this include https://learnopengl.com/Getting-started/Coordinate-Systems, https://learnopengl.com/Getting-started/Camera, https://learnopengl.com/Getting-started/Shaders, https://learnopengl.com/Lighting/Basic-Lighting.

2 Method

With most of the code completed, the modifications required were in the files main.cpp, Camera.h, phong.vs, and phong.frag. First, to view the object from the camera, the GetViewMatrix() function in Camera.h and the projection matrix in main.cpp needed to be completed and gl_Position had to be set in phong.vs. Next, to create the proper lighting effects, the ambient lighting, diffuse lighting, and specular lighting had to be set in phong.frag with additional modifications in phong.frag. Finally, with a few additional tweaks, the cube could be properly shown in the view window.

3 Implementation

3.1 View Object from the Camera

In Camera.h, all that was necessary to complete GetViewMatrix() was to return the view matrix defined using the lookAt function. The parameters required for this function are the camera position, direction, and camera up. The camera attributes “Position” and “Up” could be used directly as defined in the completed code. To get the direction, camera attributes “Position” and “Front” just needed to be added together.

Based on the provided example picture, the projection matrix in main.cpp needed to be a perspective projection matrix. This was done using the method perspective() with field of view (fov) at 45.0 degrees converted to radians, the aspect ratio calculated using the viewport width divided by its height, and the near and far plans of the frustum at 0.1 and 10.0 respectively.

With these components completed, gl_Position must be set in phong.vs as the projection matrix multiplied by the view matrix then model matrix then position vector. This order is necessary.

3.2 Write Vertex and Fragment Shaders

After the above modifications, a red cube should be visible in the viewport. To get the color and lighting right, modifications in phong.frag and phong.vs are needed. In phong.frag, to set up the ambient lighting the ambient strength must be set, then the ambient vector set by multiplying the lightColor vector by this ambient strength. Then the diffuse lighting can be set using a normal vector and light direction vector calculated by normalizing the difference between the lightPos and fragPos vectors. The dot product of the normal and light direction vectors can be used to get the diffuse impact. Multiplying this diffuse impact and lightColor results in the diffuse component.

Next, the specularStrength can be set. The view direction vector is calculated by normalizing the difference of viewPos and FragPos, and the reflect vector is calculated using the reflect() method with the parameters set as the negation of lightDir and the normal vector defined earlier for the diffuse component. Finally, the specular lighting can be calculated using the variable “spec” which is the dot product of the view and reflect directions raised to the power of 32. The specular lighting is specular strength multiplied by spec multiplied by lightColor.

Now, with these components calculated the vector “result” can be defined as (ambient + diffuse + specular) multiplied by object Color, and FragColor can be defined as vec4(result, 1.0).

3.3 Final Tweaks

With all of these changes, the cube is almost perfect, but to match the picture some adjustments still needed to be made. Changing the field of view angle to 65 degrees and changing the specular lighting variable “spec” to be raised to the power of 64 rather than 32 resulted in an image much closer to the example.

4 Results

The results are shown on the following page and in a .png file labeled ‘result’.