COSC 4370 - Homework 3

Mauricio Perez Guzman

October 2022

1 Problem

Given template files main.cpp, phong.frag, phong.vs, Camera.h, and Shader.h, the developer had to implement a 3D viewing and phong shading Model. View the object from the camera, which the cube's geometry was already given. The stubs for the shaders are provided in phong.vs and phong.frag to help implement this cube.

2 Method

Main: using the WIDTH and HEIGHT of the viewport dimensions and the camera.Zoom function we were able to implement the camera transformations.

Camera.h: Inside a GetViewMatrix we calculated what the camera should look at using the function lookAt that returns were to face the camera. The camera.h file implements the capability to move around, from the camera point of view, around an object, in our case an orange/red cube.

Phong.vs: The vertex shader manipulates the coordinates and the main point here is that it sets up the gl_Positon variable. This is used to store the current position of the current vertex.

Phong.frag: Using the light color, object color, light position, and frag position we implemented a fragment shader. A fragment shader will change the color of a particular fragment of an object.

3 Implementation

Main: using glm we were able to call on the perspective function to able to project the object onto the screen. Our perspective takes in 4 parameters the first one being the field of vision. Our field of vision was the camera.zoom because we want to see what the camera is looking at. The next one was to be able to zoom in and out we used the ratio of WIDTH over HEIGHT to be able to zoom in and out. The last two were how near and far the other objects had to be in order to be visible so basically anything that was less than 0.1 or greater than 100.0f was not visible.

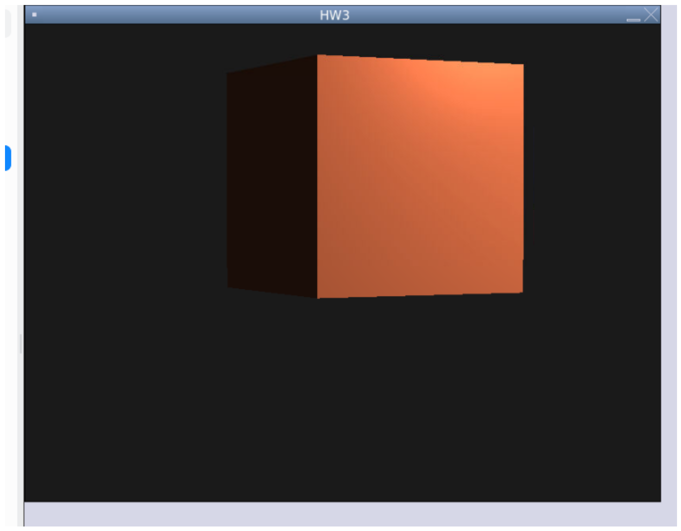
Camera.h: Inside a GetViewMatrix we calculated what the camera should look at using the function lookAt() that returns where to face the camera. The lookAt() function took in 3 parameters. The first one was the position in relation to the world. The next one told us the view we wanted to get which was the front of the cube. The third was making sure the vector was in the positive y direction which was by default set at (0.0f, 1.0f, 0.0f).

Phong.vs: gl_Position was set by multiplying the projection value by the view, model, and actual position. We also calculated the FragPosition using the view we were looking at and the model. We also calculated the normal and light position using some of the same variables.

Phong.frag: Here we set the ambient strength to 0.1. and calculated the ambient by multiplying ambient strength times the light color, this made the color less red/orange which gave us that shade on the side of the square. We diffused the object by normalizing the Normal value, setting the light direction, and calculating the diffuse by multiplying by the light color. At the end of the calculations, we were able to calculate the frag color using the result of ambient plus diffuse + specular and all times the color of the object.

Results

The output of the program was a .png file, which shows a cube that when turned has a darker side because the light isn't hitting that side. The color of the shape on the front is red/orange. The object also has a brighter spot on the top right corner of the cube which shows where exactly the light is hitting it. I had issues moving the mouse around and getting the object to display as in the instructions but after careful practice camera was easy to manipulate.



References:

https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/camera.h

https://learnopengl.com/code_viewer.php?code=lighting/basic_lighting-exercise2

https://learnopengl.com/code_viewer.php?code=lighting/materials