COSC 4370 – Assignment III

Name: Nimet Ozkan | ID: 1961233

March 2022

1 Objective

Assignment requires the implementation of 3D viewing and Phong Shader Model to shade a 3D geometric cube model. The method GetViewMatrix() in the Camera.h header file is left empty for us on purpose to implement an algorithm to view the object through the perspective of the camera. We also need to write the vertex and fragment shaders for the Phong model to shade a simple 3D cube model whose geometry is constructed in main.cpp; stubs for the shaders are provided in phong.vs and phong.frag, respectively.

2 Implementation

The 3D functions used in the assignment are provided by adding "<GL/glew.h>" from the ready-made functions in OpenGL and these functions are made available. In the Camera.h header file, we implemented glm::mat4 GetViewMatrix() to return a view matrix calculated using Eular Angles and the LookAt Matrix, so we can simulate a camera moving in the scene by moving all the objects in the scene by giving the illusion that we are moving around the camera perspective. Phong.frag and Phong.vs files have been completed to be able to design the fragment shader. Phong shading was used as a per-shard color calculation for our cube model. In the Phong.frag file, the vertex shader is provided to provide normal and position data as variables to the fragment shader to be able to use lighting system in our 3D object. Three different types of lighting used in this project: Ambient, Diffuse and Specular with normal vectors(units) that perpendicular to the surface. In the Phong.vs file we implemented the gl_Position function that holds the position of the vertex . It is a vec4 by default and want 4 floats as argument. In main.cpp we created a project matrix to provide a zoom interface using the perspective projection matrix. As a result, we got a cube with Phong shading.

2.1 Camera

OpenGl does not originally have a camera but allows us to move around in a virtual 3D scene by manipulating corner coordinates from the eyes of an imaginary camera. The only part we needed to fill in this assignment was GetViewFunction() in this Camera.h file. No need to change/add anything for the rest of the camera.h file. The main purpose of this GetViewFunction() is given as "Returns the view matrix calculated using Eular Angles and LookAt Matrix". The **LookAt** matrix is provided by OpenGL with 3 vertical axis arguments. For the current position of the camera's position vector, Front and Up, respectively. The Position vector of the camera is out of the current position, the Front vector as the target and the Up vector is already provided for us, so we could write the LookAt matrix as the ***return glm::lookAt(Position, Position + Front, Up);*** and as a result we were able to create a Forward/Front (as target) view matrix.

2.2 Phong-Shading

The Phong model makes a two-dimensional screen projection of an object appear real by describing the interaction of light with a surface in terms of surface properties and the nature of incident light. In this method, the color of each pixel on the screen is calculated separately. The intensity of a point on this surface is taken as the linear combination of these three components: Diffuse Reflection, Ambient Light, and Specular Reflection. In Phong.frag I implemented the Diffuse() function which returns this diffuse reflection as vec3. Since the position of the light is a single static variable, the lightPos is given uniformly. For diffusion we must calculate the direction vector between the light source and the position of the part, I achieved this by subtracting them among themselves. Also, I needed to normalize this light direction to put it into a unit vector. The next step was to calculate the effect of light on the existing piece by taking the dot product. The resulting value is then multiplied by the color of the light to obtain the diffuse component, resulting in a darker diffuse component the larger the angle between both vectors. In this file, I did not create a function specifically for Ambient Lighting but assigned a variable as (0.1 * lightColor) * objectColor in the main function to get some diffused light even when there is no light source. Besides that, I also implemented a Specular function to make our cube object stand out a bit. It has a similar implementation as the Diffuse() function, and it also works with the direction of the light, but it also needed to calculate and work with the eyesight direction. This is achieved by calculating the dot product between the viewing direction and the reflecting direction, and then raising it to the power of 32 (luminance emphasis value). In the main function, we need a result vector that calculates all these three lighting components of the Phong model. In the phong.vs file, I calculated the vertex position of the real part by multiplying the space coordinates as "**projection * view * model**".

3 Conclusion and Results

The output of the program was a .png file. A 3D Geometric Cube shape using Phong Shading/Lighting/Reflection. While rotating the camera using the mouse, it can also be moved using the W, A, S and D keys on the keyboard.