COSC 4370 - Homework 3

Name: Cyrus Shekari **PSID:** 2042446

March 2023

1 Objective

The program aims to implement a Phong shading model for a 3D object using the OpenGL graphics library and 3D viewing techniques. To achieve this, the program would need to calculate the view matrix and projection matrix to properly view the object from the camera's perspective. Additionally, the program would need to use vertex and fragment shaders to compute the lighting on the surface of the object based on the Phong shading model. This model considers the ambient, diffuse, and specular components of light to accurately represent the object's appearance. By combining these techniques, the program can create a realistic and visually appealing 3D object with proper shading and lighting.

2 Method

To achieve the final Phong shader model, several key components had to be implemented, including the *GetViewMatrix()* function in Camera.h, the projection matrix in main.cpp, and both the vertex and fragment shaders in phong.vs and phong.frag, respectively. The *GetViewMatrix()* function calculates the view matrix, which transforms the model's coordinates from world space to camera space. The projection matrix projects the 3D model onto a 2D screen, with *glm::perspective()* used to specify the field of view, aspect ratio, and near and far clipping planes. The vertex shader, phong.vs, calculates the position of the vertex in screen coordinates and passes the necessary variables to the fragment shader. In the fragment shader, phong.frag, lighting on the model's surface is calculated using the Phong lighting model, which involves two key functions: *Diffuse()* and *Specular()*. *Diffuse()* calculates the diffuse component of the Phong lighting model by computing the dot product of the surface normal and the light direction. *Specular()* calculates the specular component of the Phong lighting model by computing the reflected light direction and applying a glossiness factor. The final color of the model is determined by adding the ambient, diffuse, and specular components together, which is then multiplied by the object's color to obtain the desired lighting effect.

3 Implementation

The implemented program encompasses several functions and variables that work together to generate the desired Phong shading model. These functions and variables include GetViewMatrix(), *glm::perspective()*, *gl_Position*, *Diffuse()*, and *Specular()*.

3.1 GetViewMatrix

The GetViewMatrix() function returns the view matrix that transforms the world coordinates to camera coordinates. The view matrix is constructed by using the glm::lookAt() function that takes in three parameters: the camera's position, the point the camera is looking at (calculated as *Position* + *Front*), and the *Up* vector. It then returns the view matrix that represents the transformation from world space to camera space.

3.2 glm::perspective

The glm::perspective() function creates a projection matrix for the shading model. The first argument is the field of view angle in radians calculated using *camera.Zoom*. The second argument is the aspect ratio, which is the width of the viewport (*float(w)*) divided by the height of the viewport (*float(h)*). The third argument is the distance to the near clipping plane. Any objects closer than this distance will be clipped and not drawn. Finally, the last argument is the distance to the far clipping plane. Any objects farther away than this distance will also be clipped and not drawn.

3.3 gl_Position

This variable holds the position of the vertex in screen coordinates, calculated by multiplying the vertex position in the object space with the *model*, *view*, and *projection* matrices. The position of the vertex in world coordinates is calculated by multiplying the vertex position in object space with the model matrix and assigning the resulting position to the *FragPos* output variable.

3.4 Diffuse

This function computes the amount of diffuse lighting for a point on the model's surface. The *dir_light* variable is computed by subtracting the fragment position (*FragPos*) from the light position (*lightPos*) and normalizing the result. Once the direction of light is determined, the *diffusion* variable is calculated by taking the dot product of surface normal (*nm*) and the direction of the light (*dir_light*). Finally, the function returns the final diffuse component obtained by multiplying the *lightColor* by the *diffusion*.

3.5 Specular

This function computes the specular component of the lighting equation for a given fragment. The direction of the light variable (dir_light) is obtained by subtracting the fragment position (FragPos) from the light position (lightPos) and normalizing the result. Next, the reflected light direction is computed using the reflect() function and is stored in a variable named dir_reflect. Additionally, the specular intensity is computed using the dot product of the view direction (*view*) and the reflected light direction (*dir_reflect*) and raised to the power of the glossiness value (*glossiness*). Finally, the function returns the final specular component obtained by multiplying the specular intensity (*sp*) by the *lightColor* and a constant of *0.61* to adjust the intensity.

4 Results

The result of the program is a Phong shading model. This model determines the perceived color of an object by the interplay of its ambient, diffuse, and specular components. The model calculates the color of each pixel on the object's surface by summing the contributions of these components, which depend on the object's material properties and the direction and intensity of the light sources in the scene. The result is a smooth, realistic appearance of the object's surface.