COSC 4370 - Homework 3

Name: Kyle Lumbo PSID: 2170679

March 2025

1 Problem

The assignment requires the implementation of 3D viewing and the Phong shading model. The tasks involves updating the camera's view matrix and projection matrix to allow camera movement using the keyboard keys (W, A, S, D) and the rotation of the mouse. Additionally, vertex and fragment shaders need to be implemented to apply the Phong shading model to a simple orange cube. The final output should display a shaded orange cube that can be viewed from different angles by moving the camera.

2 Method

The implementation involves modifying four different files of code, each file tasked to do a specific thing

1. Main.cpp: Is the main file that initializes the OpenGL context, sets up the camera and lighting, and renders a 3D cube using the Phong shading model while handling user input for camera movement.
2. Camera.h: Defines a Camera class that manages camera position, orientation, and movement based of keyboard and mouse input.
3. Phong.frag: Implements the Phong shading model in the fragment shader, calculating ambient, diffuse, and specular lighting to shade the cube.
4. Phong.vs: Transforms vertex positions and normals to world space in the vertex shader, passing them to the fragment shader for lighting calculations.

Each file of code is used meticulously to create an OpenGL scene that implements the 3D viewing and the Phong shading model to a rendered in orange cube that has shading applied from Phong. Also allows the use of keyboard and mouse inputs to allow the user to properly display the entire cube in any view and angle.

3 Implementation

3.1 Camera Movement

The camera movement is handled by the camera class, which is defined in the Camara.h file. The class provides methods to process keyboard and mouse input, update the camera's position and orientation, and generate the view matrix.

Key Functions:

1. ProcessKeyboard: Updates the camera's position based on keyboard input (W, A, S, D keys).
2. ProcessMouseMovement: Updates the camera's orientation based on mouse movements.
3. GetViewMatrix: Generates the view matrix using the camera's position, front direction and up vector.

3.2 Phong Shading Model

The Phong shading model is implemented in the fragment shader (phong.frag). The model calculates the ambient, diffuse, and specular lighting components and combines them to produce the final color of each fragment.

Key Components:

1. Ambient Lighting: A constant light that simulates indirect lighting.
2. Diffuse Lighting: Simulates the directional impact of light on the surface, calculated using Lambert's cosine law.
3. Specular Lighting: Simulates the bright spots that appear on shiny surfaces, calculated using the Blinn-Phong model.

The vertex shader (phong.vs) transforms the vertex positions and normals to world space and passes them to the fragment shader.

4 Results

The output of the program is a 3D scene rendered in an OpenGL window. It successfully implements 3D viewing and the Phong shading model. The camera can be moved using the W, A, S, D keys and rotated using the mouse. The orange cube that is generated in the 3D scene is shaded that can be viewed from different angles, demonstrating the effectiveness

of the implemented shading model.