Foster Hare

CS 330

Professor Graham

4/19/2025

**Final Project Reflection**

**Justification**

The 3D scene setup is designed using a modular architecture through the use of classes such as SceneManager and ViewManager. These classes separate scene rendering from camera control, making the project more maintainable and scalable. Basic geometric meshes such as cubes and planes are used, which are ideal for showcasing fundamental OpenGL concepts like lighting, textures, and camera movement. These objects were chosen for their simplicity and ability to demonstrate key graphical principles.

Functionality was implemented using modular functions such as LoadSceneTextures for texture management, SetTransformations for manipulating object positioning, and SetupSceneLights for managing lighting effects. This allows for a structured and efficient scene rendering process.

**User Navigation**

User navigation is handled through the ViewManager class. Mouse movement enables control of camera orientation, while keyboard input allows navigation within the scene using standard controls such as W, A, S, and D for directional movement. These inputs update the camera’s position and orientation dynamically, offering a real-time interactive experience.

**Virtual Camera Control**

Camera control is achieved by tracking user input and updating camera properties accordingly. Mouse inputs adjust the yaw and pitch for looking around, while keyboard inputs translate the camera along its axes. The camera’s position is passed to the shader to compute lighting effects relative to the viewer’s position, enhancing realism in the scene.

**Custom Functions**

Several custom functions are used to keep the codebase modular and reusable:  
 - SetTransformations(): Applies scale, rotation, and position to objects; used for all rendered meshes.  
 - SetupSceneLights(): Configures ambient, diffuse, and specular lighting for consistent lighting setup.  
 - LoadSceneTextures(): Loads and binds texture files, separating texture logic from rendering logic.  
 - RenderScene(): Integrates object transformations, texture application, and drawing logic into one function.  
 These functions allow for clean, organized, and easily expandable code while supporting core scene functionality.

**Conclusion**

In conclusion this project had its tough moments, but finishing it felt great. OpenGL turned out to be more flexible than I thought, and the learning curve wasn’t as bad as I expected.

**Sources**

*LearnOpenGL - Transformations*. (n.d.). Learnopengl.com. <https://learnopengl.com/Getting-started/Transformations>

‌