7-1 Final Project

Armon Wilson

SNHU

CS-330 Comp Graphic and Visualization

Design Decisions

I chose to design my 3D scene around a photography theme, populating it with objects like a camera, lenses, a tripod, and film boxes. This aligns with the project's objective of creating low-polygon representations of real-world objects. The camera, constructed from cylinders for its lens and body and a box for its main compartment, exemplifies using basic shapes to form more complex objects.

Incorporating textures was key to enhancing the realism of the scene. Materials like "plastic," "metal," and "glass" were defined in the DefineObjectMaterials() function and applied to the objects in RenderScene(). This adds visual appeal and provides practical experience in using textures in 3D graphics.

The navigation scheme mirrors standard controls for first-person or free-roaming cameras in 3D environments. WASD keys control ground movement, QE keys manage vertical movement, and the mouse directs the camera's view. This control scheme is implemented in the ViewManager class, specifically within the ProcessKeyboardEvents() function, which translates keyboard input into camera actions. Mouse control is handled by the Mouse\_Position\_Callback() function, which calculates the camera's yaw and pitch based on mouse movement. Additionally, the scroll wheel adjusts the camera's speed, as managed by the scroll\_callback() function.

The functions CreateGLTexture and BindGLTextures play a crucial role in code organization and reusability. These functions simplify the process of loading, binding, and configuring textures, making the RenderScene() function cleaner and easier to understand.

The scene's lighting employs the Phong shading model, incorporating ambient, diffuse, and specular components. This is evident in the SetupSceneLights() function, where two light sources with varying properties are defined. This setup ensures the objects are well-lit and have appropriate highlights, contributing to a polished appearance.

The ability to switch between perspective and orthographic views, likely implemented by changing the projection matrix in PrepareSceneView() based on keyboard input, adds another dimension to scene visualization.

**A camera lens and a camera and a camera stick

Description automatically generated with medium confidence**A camera and a camera lens on a bed

Description automatically generated with medium confidence