Ty Wheeler

7-1 Final Project Reflection

CS 330

12/13/24

My 3D scene required many different meshes, textures and materials in order to render specific objects from my original image. For instance, the flashlight object used mostly cylinders of various sizes and used a dark plastic texture to mimic the real world object. The wallet on the other hand positioned behind the flashlight had a metal surface. I used a different metal texture and used plastic textures to render the various cards inside of the wallet. The watch similarly shared a metallic surface however, to differentiate it from the wallet I decided to use a different metal texture. The card box positioned behind the watch used a matte blue texture and the same silver texture from the watch but in a plastic texture, since the design shared the same color but not the same material.

The user can navigate the 3D scene using a combination of keyboard and mouse inputs, which control the camera's position, orientation, and speed. Keyboard controls allow movement in all directions. Using the W and S keys move the camera forward and backward, the A and D keys pan left and right, and the Q and E keys adjust vertical movement up and down. The mouse enables more precise control over the camera's orientation. As the user moves the mouse the scene view shifts accordingly to simulate looking around the environment. The scroll wheel adjusts the camera's movement speed dynamically, making navigation more flexible. The application supports toggling between perspective and orthographic projection modes using the P and O keys, allowing users to switch views based on their needs. This setup ensures an intuitive and immersive experience, providing precise control over navigation and perspective within the 3D scene.

The custom functions in the program, such as LoadSceneTextures(), SetShaderMaterial(), DefineObjectMaterials(), SetupSceneLights(), and PrepareScene(), are designed to make the code modular, reusable, and organized. For example, LoadSceneTextures() loads texture files and binds them to memory, and ensures that textures are prepared for rendering in the 3D scene. This function abstracts the texture loading process making it reusable for different scenes by simply adjusting the texture file paths and identifiers. Also the SetShaderMaterial() allows material properties to be applied to shaders, providing flexibility in reusing material settings across multiple objects. The DefineObjectMaterials() function contains the configuration of object materials making it easy to update or add new materials without modifying other parts of the program. SetupSceneLights() modularized the lights configuration, enabling reusable and adjustable light settings for various scenes. Together, these functions encapsulate specific tasks, making the codebase cleaner, easier to maintain, and adaptable for future iterations.