**7-1 Final Project Reflection**

Nathan Sakhichand

Southern New Hampshire University

CS 330

Professor Gerlach

August 17th, 2025

**Justification of Development Choices for the 3D Scene:**

In crafting this 3D scene, I chose objects that would create a realistic office desk environment, enhancing the sense of immersion and relatability. The table was represented by a plane mesh, serving as the foundation of the scene, with a texture chosen to mimic a typical wood or office desk surface, adding to the realism. The pen, composed of a cylinder for the body and a cone for the tip, was included because it's a common desk item, providing a sense of everyday functionality. Different textures for the pen body and tip differentiate between materials, like plastic and metal. An iPhone was included, modeled with a box for the body and a plane for the screen, to introduce a modern, recognizable element into the scene. The keyboard was simulated with a plane mesh and a tiled texture to replicate the repetitive pattern of keys, central to a work setting. A coffee cup, made from a cylinder for the body and a torus for the handle, was placed on the desk to represent a typical item one might find in such an environment. The notebook, using a box mesh for the cover and another for the pages, was positioned under the keyboard, adding depth and a layer of realism with different textures for each part. An 8 ball was added as a playful and visually interesting object, using a sphere mesh with a highly reflective black texture. Each object was chosen not only for its visual contribution but also for the opportunity to demonstrate different texturing techniques and material properties. Programming for the required functionality involved using custom functions to handle transformations, texturing, and material settings, ensuring modularity and ease of adjustments.

**User Navigation in the 3D Scene:**

Users can navigate the 3D scene through various input devices. The mouse is primarily used for camera orientation, allowing the user to look around by moving the cursor. This functionality would be implemented through a ‘mouse\_callback’ function, which adjusts the camera's yaw and pitch based on mouse movement, providing a first-person perspective. For movement within the scene, the keyboard would typically be used, where keys like WASD or arrow keys would move the camera in the direction it's facing. This could be handled in a ‘processInput’ function, which checks for key presses to update the camera's position. Although not implemented in the provided code, the scroll wheel could be used to adjust the field of view, simulating a zoom effect, which would be managed by a ‘scroll\_callback’ function. The camera control setup would integrate these inputs to allow for a smooth and interactive exploration of the scene, enhancing user engagement.

**Custom Functions for Modularity and Organization:**

Several custom functions were developed to ensure the code remained modular, organized, and reusable. ‘SetTransformations’ is a function that encapsulates the logic for scaling, rotating, and positioning objects in the 3D space. By calling this function with parameters for scale, rotation, and position, I can easily place and orient any object in the scene, reducing code duplication and improving readability. ‘SetShaderTexture’ simplifies texture application by binding the appropriate texture based on a tag, making it easy to switch textures for different objects without cluttering the main rendering loop. ‘SetTextureUVScale’ adjusts how textures are mapped by scaling UV coordinates, crucial for techniques like tiling, as seen with the keyboard texture, allowing for artistic control over texture appearance. ‘SetShaderMaterial’ sets up material properties like diffuse color, specular color, and shininess in the shader, which was used to define how light interacts with each object, from the matte finish of the notebook to the glossy surface of the 8 ball. These functions promote code reusability by abstracting specific operations, making the code easier to maintain and extend. Additionally, ‘SetupLighting’ and ‘SetupLightsInShader’ manage the lighting setup, ensuring that light sources are correctly integrated into the scene, which is vital for realism and depth perception. Each function contributes to a structured codebase that facilitates future modifications and enhancements.