Final Project Reflection

Rafael V Canseco

SNHU

CS-330 Comp Graphics and Visualization

Philip Enkema

12/15/24

Reflection

When designing my 3D scene, I picked objects that would not only fit the theme but also highlight different technical features. I included items like books, coffee cups, and textured materials such as wood and ceramic to create a realistic, relatable space. These choices allowed me to work with a variety of rendering techniques, like texture mapping and lighting effects, while also adding visual interest to the scene. Each object was chosen to explore different material properties, like reflections and transparency, making the scene both engaging and a good learning experience.

For the functionality, I used OpenGL with GLEW and GLFW to oversee graphics and input. I kept the code modular by splitting it into components like scene management, camera controls, and shaders, which made everything easier to manage. This structure let me focus on one feature at a time, like lighting or textures, without getting overwhelmed. Plus, it made debugging simpler and ensures I can easily add or change things later. Overall, the setup helped me efficiently implement things like object transformations and rendering while keeping the codebase clean and organized.

Navigating the scene is straightforward, with the keyboard controlling movement (like moving forward, backward, or sideways) and the mouse overseeing the camera’s orientation. Using GLFW’s input functions, I translated these inputs into camera movements through transformation matrices, so it all feels smooth and natural. I also made the camera controls adjustable, like tweaking speed and sensitivity, to make it more user-friendly. This setup gives users an easy and immersive way to explore the virtual world.

I also relied on custom functions to keep the code modular and reusable. For example, I created functions like LoadTexture() for loading and binding textures and SetupLighting() to configure the lighting in the scene. These are reusable across various parts of the program, or even in other projects, which keeps things neat and efficient. The SetView() and UpdateCamera() functions in the camera management code are another example—they manage camera positioning and direction cleanly without cluttering other parts of the program. Keeping the code modular made the complete process smoother and will make it much easier to expand or tweak later.

References

Joey de Vries. (n.d.). *Transformations*. LearnOpenGL. Retrieved December 15, 2024, from <https://learnopengl.com/Getting-started/Transformations>