Collin Mann

CS 330

7-1 Final Project Submission

When developing my 3D scene, I wanted to create a space that accurately reflected the scene I chose while also demonstrating the technical requirements of the project. The decision to model a book, a Rubik’s Cube, a green ball, and a red jar was very intentional. These objects gave me the opportunity to work with a range of primitive shapes and combinations. The book was built using a scaled box, which allowed me to capture the rectangular dimensions of a paperback novel. The Rubik’s Cube, although also a box, served as a useful example of reusing the same primitive with different scaling, positioning, and eventually a different texture. The green ball was created with a sphere, which contrasted well with the flat edges of the other models and introduced more geometric variety. Finally, the jar required a combination of a cylinder for the base, a cone for the lid, and a sphere for the knob. By combining these primitives, I was able to create a more complex shape, which helped in using multiple primitives in at least one object.

Throughout this process, I paid attention to proportion and placement. I carefully scaled and positioned each object so that they matched as closely as possible to the real-world scene in the photos. The book was scaled wider and flatter to reflect its dimensions, the Rubik’s Cube was placed on top of it to appear as if it was sitting naturally, and the ball was adjusted so that it appeared slightly smaller and positioned in front of the jar, just as it was in the images. This process gave me a chance to think about how to translate real-world scale into a digital environment while still working within the constraints of low-polygon primitives.

Navigation of the scene was also a key part of the development. I implemented camera controls that give the user the ability to move through the space fluidly. The keyboard allows the user to pan left and right, move forward and backward, and adjust vertical position up and down. The mouse enables orbiting around the scene, allowing the user to view the objects from any angle, while the scroll wheel gives precise control over zooming in and out. These controls work together to create an intuitive way for someone to explore the 3D environment and examine each object in detail.

To ensure my program was modular and organized, I relied on a number of reusable functions. The function SetTransformations() was particularly important because it standardized how scaling, rotation, and translation were applied to each object. Rather than rewriting transformation logic for every model, I could simply call this function with the parameters I needed. Similarly, SetShaderMaterial() and SetShaderTexture() allowed me to assign materials and textures without embedding that logic into the rendering of each object. This approach keeps the rendering code clean and makes it much easier to swap textures or change material properties later. Functions like DefineObjectMaterials() and PrepareScene() also helped me keep the setup process centralized, meaning that materials, meshes, and textures could be defined once and then referenced consistently throughout the project.

Overall, the choices I made in developing this scene were guided by both the requirements of the project and the desire to create a faithful digital representation of the scene I chose. By working with simple primitives and combining them where necessary, I was able to replicate real-world objects while keeping polygon counts low. By implementing flexible navigation and modular code, I was able to build a scene that is both functional for the user and maintainable for me as a developer. This project reinforced the importance of planning object construction around primitives, thinking critically about scene layout, and writing reusable code that makes the development process more efficient.