Daniel Schmidt

CS – 330 Comp Graphic and Visualization

7-1 Final Project Submission

***Reflection: Design Decisions***

In developing my 3D scene, I selected a variety of objects—spheres, cubes, pyramids and cones because they provide a good balance of geometric diversity. The spheres highlight how textures wrap smoothly around curved surfaces, while the cube demonstrates even tiling across flat faces. The pyramids and cones gave me the chance to work with triangular sides, which helped in showing how textures can be applied to non-square surfaces. I also included a textured ground plane so that the objects appear anchored in space, creating a more complete and realistic environment. These choices were intentional in order to showcase ***lighting, texturing, and transformations*** on a range of surfaces.

Navigation within the 3D scene is handled through keyboard and mouse inputs. I set up the virtual camera to respond to the ***W, A, S***, and***D*** keys for moving forward, left, backward, and right, and to the ***Q*** and ***E*** keys for moving vertically up and down. The ***mouse controls*** the camera’s yaw and pitch, allowing the user to freely look around, while the scroll wheel adjusts the zoom level. This configuration provides an intuitive and flexible way to explore the scene, similar to free-fly controls found in many 3D applications. The camera system updates the view matrix dynamically, ensuring smooth movement and consistent orientation while navigating the objects.

To keep my program modular and organized, I created several custom functions. The ***SetupSceneLights()*** function manages all lighting configurations, making it easy to adjust the ambient, diffuse, and specular values in a single place. The ***LoadTexture()*** function is reusable for importing and binding different images to objects without duplicating texture setup code. The ***RenderObject()*** function applies translations, rotations, and scaling before drawing objects, which allows me to reuse it for any shape in the scene. Finally, ***ProcessInput()*** centralizes how the program handles user input, keeping the main loop clean and easy to follow. By structuring my code this way, I made it easier to expand the project, debug issues, and maintain clear separation between scene setup, rendering, and interaction.

Overall, these development choices demonstrate how textures, lighting, and navigation can be combined to create an interactive 3D environment. The use of modular functions not only simplifies the code but also makes the project extensible for future enhancements.

A group of objects on a surface

AI-generated content may be incorrect. A group of geometric shapes

AI-generated content may be incorrect.