Rene Guzman

Southern New Hampshire University

CS-330 Comp Graphic and Visualization

Reflection on Final 3D Scene Development

Justification of Development Choices

For this 3D scene, I created a kitchen shelf display featuring a plant in a pot, a stacked bowl set, a wooden mortar and pestle, a golden torus sculpture, and a backdrop wall—all designed using basic geometric shapes like toruses, spheres, cubes, and planes. I chose these objects to reflect both artistic and technical intent: the torus sculpture acts as a visually dynamic centerpiece that demonstrates lighting and texture reflection; the plant and kitchen items provide realistic, recognizable forms that align with the theme of a still-life interior.

Each object was selected with the goal of testing and displaying various OpenGL features, including material shading, texture mapping, and lighting models. These items also allowed me to apply different material properties—such as metallic reflection for the torus, brushed steel for the bowls, rustic textures for the mortar and pestle, and leaf-like textures for the plant. These choices ensured I could showcase how different surface properties interact under a dynamic lighting setup using the Phong lighting model. The scene also required successful integration of camera movement, user navigation, and texture loading, all of which I accomplished using modular and object-oriented C++ code.

Scene Navigation and Camera Setup

User navigation through the scene is implemented with full keyboard and mouse input using the GLFW library. The scene allows users to move through the virtual space using W, A, S, D keys to move forward, left, backward, and right respectively. Additionally, Q and E keys let the user move the camera vertically. The mouse is used to control the viewing direction, and the scroll wheel adjusts the speed of movement. This interaction design allows the user to intuitively orbit, zoom, and pan around the entire scene, enabling close inspection of individual objects and their textures under lighting.

To handle this functionality, I created a ViewManager class that manages camera logic and user input. The camera's movement is controlled by calculating delta time for smooth transitions and using directional input to update the camera's position and orientation vectors. I also added functionality to toggle between orthographic and perspective projections using the O and P keys. This allows the user to experience the scene in both styles of 3D viewing, enhancing usability.

Custom Functions for Modularity and Reuse

Several custom functions were developed to ensure modularity and clarity in the codebase. In the SceneManager class, I created the following reusable methods:

* SetTransformations(): Takes in parameters for position, rotation, and scale, and applies them in a consistent transformation order. This function ensures each object can be positioned and oriented uniquely without hardcoding transformation matrices.
* SetShaderTexture(std::string tag): Allows the assignment of a texture to any object by using a texture tag, reducing duplication and enabling quick switching of texture assets.
* SetShaderMaterial(std::string tag): Applies ambient, diffuse, and specular material properties based on the object's tag. This supports easy reuse across multiple objects that share the same material characteristics.
* SetupSceneLights(): Abstracts light position and property setup into one method that initializes four light sources, each with its own ambient, diffuse, and specular intensities.
* PrepareScene(): Combines texture loading, material definition, and light setup into a single setup routine, ensuring that each component of the scene is initialized in one place.

Each of these methods contributes to reducing code repetition and enhancing the maintainability of the project. The modular structure made it easy to adjust parameters during testing, such as fixing lighting brightness or changing object materials. Overall, these decisions enabled a flexible and visually effective 3D scene implementation.