Final Project Design Decisions

Tyler Daniels

Southern New Hampshire University

Course Number: CS-330

Professor Graham

02-23-25

Final Project Design Decisions

The objects included in the 3D scene were chosen based on their relevance to creating a realistic and interactive environment. The **SceneManager** class is responsible for managing these objects, handling their textures, materials, and transformations. Each object was selected to demonstrate different rendering techniques such as texturing, lighting, and shading. To efficiently manage rendering performance, textures are stored in an array (**m\_textureIDs**). The DefineObjectMaterials() function allows for defining different material properties, ensuring that each object has a realistic appearance.

The program ensures that the required functionality, such as rendering, transformations, and lighting, is implemented in a modular fashion. Functions like SetTransformations() and SetShaderMaterial() allow for dynamic configuration of objects, making it possible to adjust their properties without hardcoding values. The **SceneManager** class also manages shader interactions, ensuring that materials and textures are properly applied.

User navigation within the 3D scene is handled through both keyboard and mouse inputs, with the **ViewManager** class responsible for managing camera movement and user interactions. Keyboard controls allow movement through **W/S** for forward and backward, **A/D** for left and right, and **Q/E** for upward and downward movement. The **P** and **O** keys toggle between perspective and orthographic projection modes, while the **ESC** key closes the application. Mouse movement dynamically changes the camera orientation, ensuring smooth transitions. The Mouse\_Position\_Callback() function processes mouse input to adjust the camera view accordingly.

The **ViewManager** class initializes and controls the virtual camera using the **Camera** class, starting at a predefined position of **(0.0f, 5.0f, 12.0f)** while looking downward at an angle. The camera supports both **perspective and orthographic projections**, which can be toggled using the **P** and **O** keys. Real-time control is achieved through ProcessKeyboardEvents(), while smooth camera rotation is handled by processing mouse movements via ProcessMouseMovement(xOffset, yOffset), ensuring a fluid and interactive navigation experience.

To maintain a modular and organized code structure, several custom functions were implemented. The PrepareSceneView() function in **ViewManager** converts 3D object space into a 2D viewport projection, captures user input dynamically, and configures the view and projection matrices for rendering. The PrepareScene() function in **SceneManager** sets up objects, textures, and lighting before rendering begins, ensuring the scene is properly configured. The RenderScene() function handles the actual rendering process, ensuring objects are displayed with the correct transformations and materials. The CreateDisplayWindow() function in **ViewManager** initializes the OpenGL display window, manages input handling, mouse capture, and transparency settings. Finally, the ProcessKeyboardEvents() function processes user input to move the camera and switch between projection modes.

By structuring the code in this way, the system remains **scalable** and adaptable, allowing for easy feature additions or modifications without requiring significant changes to the existing codebase. This approach ensures a well-organized, efficient, and interactive 3D scene management system.