Southern New Hampshire University  
CS 330 - Comp Graphic and Visualization

Mohamed Elhassan

**Justification of Development Choices for the 3D Scene**

The 3D scene was designed to replicate a desk workspace, inspired by its familiarity and suitability for showcasing technical features. Objects like the monitor, keyboard, table, legs, and additional details such as keys and a monitor stand were selected to create a cohesive and realistic environment. These objects were constructed using basic shapes like boxes, cylinders, and planes, as seen in the code within RenderScene() where functions like DrawBoxMesh and DrawCylinderMesh render these elements.

This scene setup provided opportunities to implement textures, such as applying a marble texture to the table or a static texture to the monitor, enhancing realism. The lighting setup, including directional and point lights, added depth and detail to the scene, as configured in PrepareScene().

User Navigation in the 3D Scene

The navigation system uses the keyboard and mouse for intuitive interaction:

* **Keyboard**: Keys such as WASD or arrow keys control forward, backward, left, and right movements. Q and E keys allow vertical motion for a full 3D navigation experience.
* **Mouse**: Mouse movements enable rotation, while the scroll wheel manages the speed at which the camera moves.

The camera’s behavior is defined through the Camera class, with methods like ProcessKeyboard and ProcessMouseMovement updating its position and orientation dynamically. The Mouse\_Position\_Callback and Scroll\_Callback functions handle real-time adjustments based on user input, ensuring smooth navigation. The program supports both perspective and orthographic projections, toggled with the P and O keys, as demonstrated in PrepareSceneView().

**Custom Functions for Organized Code**

Several custom functions and classes were implemented to keep the code modular and efficient:

1. **CreateGLTexture(filename, tag)**:
   * Loads and configures textures from image files, as shown in PrepareScene() for objects like the table and monitor.
   * Ensures textures are easily reusable across different parts of the scene.
2. **SetTransformations(scaleXYZ, XrotationDegrees, YrotationDegrees, ZrotationDegrees, positionXYZ)**:
   * Handles object scaling, rotation, and positioning. For example, this function is used extensively in RenderScene() to place and size components like the table, monitor, and keyboard.
3. **PrepareScene()**:
   * Initializes the scene by loading meshes, textures, and materials and configuring lighting. It ensures all elements are ready for rendering.
4. **RenderScene()**:
   * Iterates through and renders all objects in the scene using modularized transformation and texturing processes.
5. **Camera Management Methods**:
   * Methods like ProcessKeyboard and ProcessMouseMovement manage the camera’s movement and orientation, enabling smooth navigation based on user input.

By leveraging these functions and a clean structure, the project efficiently integrates textures, transformations, and lighting. The modularity ensures that components can be easily updated or reused, making the codebase flexible for future enhancements.