Rajya Kolluri

Final Project Reflection

Justification for Development Choices

**Selection of Objects**

* Table: The table serves as the foundational object in the scene, providing context and spatial orientation. It represents a workspace, which is a familiar environment for many users, making the scene relatable and functional.
* Laptop: The laptop is a central object, symbolizing productivity and modern workspaces. It also allows for interactive elements, like turning the screen on/off, opening/closing the laptop, which can be programmed to enhance user engagement.
* Cup: The cup adds a touch of realism and homeliness to the scene. It can be used to showcase reflective properties and surface textures in 3D rendering.
* Pencils: Pencils scattered or placed in a holder add to the workspace's authenticity and detail, contributing to a more immersive environment.
* Books: Books provide texture variety and additional elements for interaction. They also enhance the scene’s educational or professional theme.
* Photo Frames: Photo frames personalize the workspace, making it feel lived-in. They can also be used to demonstrate image loading and texture mapping.
* Lamp: The lamp serves both a practical and aesthetic purpose. It can act as a light source, demonstrating lighting effects and shadows, and can also be an interactive element where users can toggle the light on and off.

**Programming for Required Functionality**

* Object Interaction: Each object in the scene is programmed with properties that allow for interaction. For instance, clicking on the laptop can toggle its screen state, or turning the lamp on/off changes the lighting dynamically.
* Physics and Collision Detection: Basic physics are applied to ensure objects stay on the table and do not intersect unrealistically. Collision detection is used for user interactions.
* Lighting: Different lighting setups (ambient, point, directional) are used to highlight the objects and create realistic shadows and reflections.
* Texturing: Textures are applied to objects to enhance their realism, such as wood grain on the table or a glossy finish on the photo frames.

**User Navigation in the 3D Scene**

* Keyboard Controls: Users can navigate the scene using the WASD keys for forward, backward, and lateral movement. The arrow keys or Q/E can be used for looking around and adjusting the camera angle. For views, 1- front view, 2- side view, 3- top view, 4- perspective view.
* Mouse Controls: Mouse movement controls the camera's rotation, allowing users to look around the scene fluidly. Left-click can be used for interacting with objects.

**Setting Up Virtual Camera Controls**

* Initialization: The camera is initialized at a certain position and orientation within the scene, typically set to a starting viewpoint that offers a comprehensive overview of the workspace.
* Camera Movement Functions: Functions are created to handle camera movements, such as moveForward(), moveBackward(), strafeLeft(), strafeRight(), and rotateCamera().
* Input Handling: Input handling functions map keyboard, mouse, and gamepad inputs to the corresponding camera movement functions. For instance, pressing the 'W' key triggers the moveForward() function.
* Smooth Transitions: Implementing smooth transitions and dampening effects to make the camera movement and rotation feel natural and responsive.

**Custom Functions for Modularity and Organization**

* initializeScene(): Sets up the scene with all objects, their initial positions, and properties. This function can be reused to reset the scene or switch between different scenes.
* renderScene(): Responsible for drawing all objects in the scene each frame. It ensures that all elements are consistently rendered and updated.
* updateLighting(): Adjusts lighting parameters based on the time of day or user interactions, such as turning the lamp on or off.
* handleInput(): Centralizes input processing, making it easier to manage different types of input devices and ensure they all interact with the camera and objects correctly.
* toggleLaptopScreen(): Toggles the state of the laptop screen, demonstrating how individual objects can have specific interactive functions.
* applyTexture(object, textureFile): A reusable function to apply textures to objects, promoting code reuse and simplicity when adding new textured objects to the scene.

These functions encapsulate specific tasks, making the codebase easier to manage, understand, and extend. By keeping functions focused and reusable, it's simpler to introduce new features or modify existing ones without affecting other parts of the program.

*References:*

Angel, E., & Shreiner, D. (2014). Interactive Computer Graphics: A Top-Down Approach with WebGL (7th ed.). Pearson.

Akenine-Möller, T., Haines, E., & Hoffman, N. (2018). Real-Time Rendering (4th ed.). CRC Press.