Final Project

A phone and a box of tissues on a table

AI-generated content may be incorrect.

In developing my 3D scene, I drew inspiration from a selected photograph depicting a modern tabletop setup, aiming to replicate its components with low-polygon 3D representations for efficiency and realism. My choices for objects— a wooden table, gold vase, glass cup, tissue box, and smartphone—were justified by their alignment with the image's layout and the need to incorporate diverse primitive shapes as required. For instance, the table was modeled as a simple box primitive, scaled to 30x2x15 units, to serve as the foundational surface, minimizing complexity under 1,000 triangles while providing a broad plane for other objects. This choice emphasized practicality, as boxes are versatile for flat surfaces and easy to texture with a wood image for authenticity. The vase combined a sphere for the body and a torus for the opening, allowing me to demonstrate compound shapes; the sphere's smooth curvature captured the vase's rounded form, and the torus added a flared rim, justifying the use of multiple primitives to achieve a more detailed yet low-poly model. Similarly, the glass used a tapered cylinder to mimic its conical shape, which was ideal for representing transparency and height variation without excessive polygons. The tissue box and phone reverted to box primitives for their rectangular forms, ensuring variety while keeping the scene cohesive. These selections balanced simplicity with visual fidelity, adhering to the rubric's demand for at least four objects and multiple basic shapes like box, sphere, torus, and tapered cylinder.

Texturing and lighting choices further polished the scene, driven by the goal of realism and performance. I sourced royalty-free 1024x1024 textures, such as wood.jpg for the table and gold-seamless-texture.jpg for the vase, projecting them accurately with UV scaling (e.g., 2.0x1.0 for the table) to avoid distortion. This enhanced depth, making the objects appear lifelike without high computational cost. Lighting incorporated a directional light for broad illumination and a colored point light (warm orange-red) to highlight details, using Phong shading's ambient, diffuse, and specular components for nuanced reflections—e.g., high shininess (80.0f) on the phone for a glossy finish. Positioning objects via X, Y, Z coordinates (e.g., vase at -6.0f, 0.925f, 2.5f) mirrored the photo's arrangement, with overlaps resolved by incremental adjustments to create a natural composition.

User navigation in the 3D scene is intuitive and immersive, leveraging keyboard and mouse inputs to control the virtual camera effectively. The camera orbits the scene with a sufficient radius to capture all objects, ensuring comprehensive views. Horizontal and vertical movement uses WASD keys for forward/backward (W/S) and left/right (A/D) panning along the X and Y axes, while QE keys handle upward/downward traversal on the Z-axis, allowing users to explore depth seamlessly. This setup provides precise control, with the camera's speed adjustable via mouse scroll—scrolling up increases speed for faster navigation, down decreases it for detailed inspection, clamped between 1.0f and 50.0f for usability. Orientation changes, independent of position, are managed by mouse cursor movement: dragging adjusts pitch (up/down) and yaw (left/right), enabling users to look around without relocating the camera. For example, pitching upward reveals object tops, while yawing scans the tabletop laterally. Depth navigation integrates with these, as zooming (via W/S) brings users closer or farther, maintaining light projection on all elements. Additionally, pressing 'O' toggles to orthographic (2D) view for flat analysis, and 'P' returns to perspective (3D) for immersive depth, switching matrices while preserving orientation. This system empowers users to interact dynamically, matching real-world exploration.

To enhance modularity and organization, I implemented custom functions that break down the program into reusable, logical units, promoting readability and maintainability. In SceneManager.cpp, functions like RenderTable(), RenderVase(), and similar for each object encapsulate transformations, texturing, and drawing for specific models—e.g., RenderVase() handles sphere and torus rendering with shared SetTransformations() and SetShaderTexture() calls, reusable across objects by passing parameters like scale and position. This modularity allows easy additions or modifications without altering core logic. SetTransformations() itself is a key custom function, computing model matrices from scale, rotation, and position inputs, then setting them in the shader; it's called before every draw, making transformations reusable and centralized. Similarly, SetupSceneLights() configures lights once, reusable for the entire scene, while DefineObjectMaterials() defines and stores materials (e.g., gold with specular values) for quick retrieval via FindMaterial(). In ViewManager.cpp, ProcessKeyboardEvents() modularizes input handling, checking keys like WASD/QE for movement and O/P for projection toggling, keeping the main loop clean. PrepareSceneView() computes view/projection matrices and updates shaders, reusable per frame. These functions reduce redundancy—e.g., material and light setups are defined once and applied globally—ensuring the code is organized, scalable, and adheres to functional logic best practices.