Ryan Orton

CS320

08/17/25

**Final Project Reflection**

For this project I chose a simple holiday scene because it maps cleanly to low-poly primitives while still exercising texturing, lighting, and camera control. The tree is a composite object made from a cylinder trunk, three stacked cones for foliage, and several spheres as ornaments, all placed on a plane floor. Keeping everything to basic meshes guarantees well-spaced triangles and a total polygon count comfortably under the 1,000-triangle guideline. I textured two objects: the cylinder with a wood.jpg and the cones with a pine.jpg using high-resolution, royalty-free images. I enabled mipmapping and exposed a UV scale so texel density can be tuned without altering geometry. Lighting is set up to demonstrate the full Phong model: an overhead “sun” point light above the tree establishes strong diffuse and specular on the foliage, a cool-tinted colored fill satisfies the requirement for a colored source and softens shadows, a subtle rim/back light separates silhouettes, and a gentle ground-bounce light prevents the underside from going completely dark. Textures are loaded via relative paths in Utilities\textures\ so the scene works when zipped and shared.

Users can navigate the scene with both translation and orientation controls. WASD moves the camera forward/back and left/right on the XZ plane, while Q and E move vertically along Y. Mouse movement adjusts yaw and pitch to look around, with pitch clamped to avoid flipping; the mouse scroll wheel adjusts movement speed on the fly so it’s easy to switch between broad orbits and close inspection. Pressing P toggles between perspective and orthographic projections while preserving the camera’s current orientation, which is useful for comparing form and proportion. I also started the camera far enough back to frame the full tree and floor, and the multi-light setup ensures that as the user moves around, no part of the composite object falls into complete shadow.

To keep the code modular and maintainable, I separated preparation from rendering and isolated repeated tasks into small, reusable helpers. PrepareScene() loads meshes once, defines all materials (ambient, diffuse, specular, shininess), configures the four lights, and loads/binds textures by string tag; RenderScene() handles per-frame transforms and draw calls only. Helper functions such as CreateGLTexture() and BindGLTextures() encapsulate image loading, filtering, and mipmap generation; FindTextureSlot()/FindTextureID() and FindMaterial() provide clean lookups so the draw code can reference tags instead of raw OpenGL handles. SetTransformations() centralizes the scale/rotate/translate model matrix in a consistent order, and SetShaderColor(), SetShaderTexture(), SetTextureUVScale(), and SetShaderMaterial() set only the uniforms needed for each object, keeping the render path short and readable. Camera math and input handling live separately in a small camera struct with processInput, mouse, and scroll callbacks, making those controls reusable in other scenes without touching rendering logic. Collectively, these choices produce a clear, testable structure that meets the project requirements while remaining easy to extend.