**CS 330 Final Project**

Re-creating a scene with nothing but OpenGL primitives forced me to choose objects that balanced visual interest against technical practicality. I began the semester wanting a mouse, an Xbox controller, a monitor, and even a little starship model. After wrestling with those shapes for weeks, I reshaped that list to a laptop, a pencil cup with three pencils, a small book stack, and a low-poly mouse. Every one of those items can be broken down into cylinders, tapered cylinders, boxes, or planes, shapes already supported by the ShapeMeshes helper. That choice let me spend time on placement, scale, and interaction instead of writing new mesh loaders. Textures were kept intentionally spare, the floor and wall use simple wood-and-tile bitmaps to ground the scene, while most objects rely on flat colors so their silhouettes read clearly when lit. For my lighting a top-down key light gives strong highlights, a camera-aligned fill light prevents pitch-black shadows, and a muted grey ambient term (light source 2) keeps the overall exposure, even though I still consider the result more functional than cinematic.

A user can explore that layout freely. I wired the camera class to the familiar W-A-S-D keys for forward, left, back, and right translation; Q and E raise or lower the eye point. The mouse yaw-pitches the view so objects can be examined from any angle, and the scroll-wheel changes movement speed for fine positioning. Pressing O toggles the projection matrix between perspective and orthographic. I found that feature handy for sanity checks, perspective mode shows depth during normal use, while orthographic mode reveals subtle alignment errors when I’m lining up the hinge cylinders against the laptop base, or the scene to the ground.

Keeping the code maintainable was the other big challenge, so I leaned hard on small, reusable helpers. SetTransformations() composes translation, three-axis rotation, and scale into a single model matrix; because every render routine calls it, I can tweak object placement by adjusting one call site instead of hunting through shader code. SetShaderColor() and SetShaderTexture() hide the uniform plumbing needed to flip between solid-color and textured rendering, and SetTextureUVScale() lets me tile the wood floor without touching the fragment shader. At a higher level, each major object lives in its own Render…() function—RenderLaptop, RenderBookStack, RenderPencilCup, and so on, so swapping their positions or scales became a matter of editing a handful of constants. Finally, SetupSceneLights() encapsulates every light-source uniform. My design choices being simpler primitives, minimal but effective texturing, light Phong setup, full six-degree camera navigation, and a library of one-purpose helper functions made for a decent scene.