When I started building my 3D scene, I wanted to replicate something realistic and meaningful—something that could reflect the kind of workspace I personally relate to. That’s why I chose to recreate a modern office desk setup based on a specific reference image. The scene includes everyday items like a notebook, a pen, a few stacked binders, a coffee cup, a pencil holder, a piece of paper, and a tablet. All of these objects were not only familiar but also allowed me to work with multiple basic 3D primitives such as boxes, cones, cylinders, and planes. These objects were also ideal for demonstrating required functionality—such as applying textures to curved and flat surfaces, combining multiple shapes into one object, and experimenting with lighting and shading effects that respond to different materials.

In terms of development choices, I focused on low-polygon modeling for performance, while relying on high-quality textures for realism. The notebook, for instance, was designed as a composite object made from multiple shapes to mimic a real-world notepad with pages, a cover, and a metal clip. The pen uses a combination of cone and cylinder meshes to create a realistic tapered tip. These choices reflect a balance between visual clarity and rendering performance, keeping polygon counts low while preserving recognizable forms. Each object was positioned thoughtfully using 3D coordinates, closely matching the layout of the original reference image. To bring the scene to life, I projected textures carefully onto key surfaces such as a wood grain texture for the desk and detailed images for the notebook and paper, using high-resolution images that enhanced visual fidelity. I carefully positioned and scaled these textured objects to match the reference image as closely as possible, paying special attention to spacing and alignment to make everything feel natural.

To navigate and make my 3D scene fully interactive, I implemented robust camera navigation controls. Users can explore the scene with fluid movement on all three axes—forward and backward using W and S, side to side with A and D, and vertical motion using Q and E. This was achieved by mapping these keys to camera translation logic within ProcessKeyboardEvents(). For orientation, the mouse controls camera rotation dynamically through pitch and yaw, captured via Mouse\_Position\_Callback(). This simulates a natural viewing experience similar to first-person exploration. Additionally, I added scroll wheel input to let users fine-tune movement speed on the fly, creating a more responsive and customizable interaction. To further enhance the scene’s usability and visual analysis, I implemented projection switching. By pressing P, users can enter perspective mode, which gives a more immersive 3D view. Pressing O switches to orthographic mode, repositioning the camera to a top-down view ideal for reviewing object alignment and spatial relationships. This dual-mode view adds functional value to the scene and showcases my understanding of projection matrices in OpenGL.

Underlying all of this, the organization of my code relied on custom modular functions that kept the program clean and maintainable. DefineObjectMaterials() is used to manage all object-specific material properties in one place, separating visual characteristics from logic. SetupSceneLights() handles the configuration of point lights and other light sources, allowing me to easily adjust lighting without touching unrelated parts of the code. RenderScene() drives the entire rendering pipeline, relying on repeatable calls like SetTransformations() and SetShaderTexture() to consistently apply changes across objects. Because each function is self-contained and flexible, I can easily reuse them in future projects by simply passing new parameters or plugging in different objects, without needing to rewrite core rendering or lighting logic.

Overall, this project reflects my ability to translate a real-world reference into a functional, navigable 3D environment. I combined visual fidelity with efficient modeling and interactive controls, all backed by clean, modular code practices. The experience strengthened my grasp of OpenGL concepts while reinforcing the value of design thinking in technical execution.