Reflection on 3D Scene Development and Navigation

When I set out to design my 3D scene, I wanted to craft an environment that was both inviting and technically illustrative. I decided on a café setting featuring a wooden table, coffee cups, a vase of flowers, and a hanging lamp—four elements that convey a cozy atmosphere while allowing me to showcase fundamental 3D modeling and rendering techniques. Each object was chosen because it offers distinct shapes and materials: the table (a sturdy geometric form), the coffee cups (smaller, more detailed objects), the vase of flowers (introducing an organic touch), and the hanging lamp (overhead interest and a light source). These objects collectively reflect the essence of a relaxed gathering place while fulfilling the need to experiment with multiple primitive shapes in 3D.

From a programming standpoint, I addressed the core requirements through careful planning and modular design. First, I created low-polygon versions of each object to optimize performance and ensure shorter rendering times. By using basic primitives—cylinders for table legs or vase bodies, boxes for table tops, and spheres for flower buds—I kept the polygon count manageable. Implementing textures (such as wood grain for the table and porcelain for the cups) allowed me to introduce surface variety without complicating the geometry. I employed the Phong lighting model to accentuate the different materials, focusing on ambient, diffuse, and specular components for a balanced, realistic look. This lighting setup ensures that the user can visually differentiate metal from wood, for instance, based on how each surface reflects light.

Navigation and Camera Controls

To ensure an intuitive user experience, I implemented a free-fly camera system driven by keyboard and mouse inputs. Pressing **W** moves the camera forward, **S** moves it backward, **A** shifts it left, and **D** shifts it right. In addition, **Q** and **E** allow vertical motion along the Y-axis. These mappings mirror conventional first-person navigation schemes, making the controls recognizable to anyone familiar with 3D applications.

The mouse controls the camera's orientation. Moving the mouse horizontally adjusts the yaw angle, letting me look left or right, while vertical mouse movement changes the pitch angle, enabling me to look up or down. The scroll wheel adjusts the camera's movement speed, providing the user with fine control over navigation. When small, precise movements are needed—such as examining details on the tabletop—lower speeds can be used. Conversely, a higher speed is helpful for zooming out to survey the entire scene.

Moreover, I allow the user to press **P** and **O** to toggle between perspective and orthographic projections. In perspective mode, objects in the distance appear smaller, creating a sense of depth. Orthographic mode, on the other hand, displays objects at consistent scales regardless of their

distance from the camera. This dual-projection approach helps showcase how different mathematical models can alter the perception of space in a 3D environment.

Modularity and Custom Functions

A key principle in my development process was keeping the code modular and readable. I created separate classes—such as **SceneManager** for loading and drawing objects, and **ViewManager** for handling camera controls and projection logic. Within **SceneManager**, I have functions like RenderTableTop(), RenderCoffeeCup(), and RenderVase() for drawing each object. These methods encapsulate the specific transformations, materials, and textures needed for each object type. This means if I wanted to add another decorative piece or remove an existing one, I could do so without overhauling the entire rendering pipeline.

Similarly, **ViewManager** handles camera behavior, user inputs, and projection toggling. By delegating these tasks to specialized functions such as ProcessKeyboardEvents() and PrepareSceneView(), I ensure that each logical step—like reading keyboard input or updating the projection matrix—stays well organized. The benefits of this approach are twofold: it simplifies the debugging process, and it facilitates future extensions or collaborations, as each function's purpose is clear and self-contained.

Overall, my design decisions stem from a desire to balance visual appeal with practical demonstration of 3D rendering, lighting, and navigation principles. By choosing a small set of objects characteristic of a cozy café scene, I was able to focus on solidifying each step of the pipeline, from basic polygon modeling to camera interactivity. This modular approach will serve as a strong foundation if I decide to expand the scene later, add new objects, or adjust the lighting and materials to create different environments.