Design Decisions for 3D Desk Scene

CS 330: Computational Graphics and Visualization

# 1. Development Choices and Object Selection

I chose to recreate my actual desk setup because it gave me familiar objects to model while demonstrating different OpenGL techniques. The scene includes a monitor, keyboard, mouse, desk lamp, coffee mug, and two stacked books. Each object was selected to show a different modeling approach using basic geometric shapes.

The monitor uses box meshes to create the screen and stand, showing how rectangular shapes work with texture mapping. The coffee mug combines multiple cylinder meshes for the body and handle, demonstrating how complex objects can be built from simple shapes positioned at different angles. The desk lamp shows hierarchical modeling by connecting a base, two arms, and a cone shaped shade into one complete object.

I added two stacked books to the right of the coffee mug to add more color and visual interest to the scene. The bottom book is green and the top book is red, and I rotated them at different angles to make the arrangement look more like the set up in the photo. Programming these objects required using transformation matrices for positioning and the SetTransformations function to handle scale, rotation, and position for each object. The material system lets me apply different surface properties like plastic for the books and metal for the lamp to make objects look more realistic under lighting.

# 2. Scene Navigation and Camera Controls

Users can navigate the 3D scene using standard keyboard and mouse controls. The WASD keys control movement where W moves forward, S moves backward, A moves left, and D moves right. The Q and E keys let users move up and down to see the scene from different heights. This gives complete freedom to explore the entire workspace from any angle.

The mouse controls where the camera is looking. Moving the mouse left and right rotates the view horizontally to pan around the scene. Moving the mouse up and down tilts the camera to look up at the lamp or down at the desk surface. The scroll wheel adjusts how fast the camera moves, which is helpful for both quick navigation and careful examination of specific objects.

I set up the camera system to be framerate independent so movement stays smooth and consistent regardless of computer performance. The controls follow standard conventions used in most 3D applications, making the scene easy to navigate without needing special instructions.

# 3. Custom Functions for Code Organization

I created several custom functions to keep the code organized and avoid repetition. The most important one is SetTransformations which handles all the math for positioning objects in 3D space. Instead of manually writing transformation code for every object, I just call this function with the scale, rotation angles, and position.

The SetShaderMaterial function manages how objects react to light by setting up material properties like shininess and reflectivity. I can pass in a material name like plastic, metal, or rubber and it applies the right properties automatically. This means I do not have to remember or manually set all the material values each time. The SetShaderColor function applies solid colors to objects, which I used for the green and red books. The SetShaderTexture function applies image textures like the one I used for the monitor screen.

These functions make my code reusable because I can use the same transformation and material systems in other projects. If I need to change how materials work, I only have to update the function in one place instead of searching through hundreds of lines of code. The functions also make debugging easier because each one has a clear purpose. For example, if object positioning is wrong, I know to check SetTransformations. If lighting looks off, I check SetShaderMaterial.

The mesh drawing functions like DrawBoxMesh, DrawCylinderMesh, and DrawSphereMesh hide the complex OpenGL rendering commands behind simple function calls. This organization follows good programming practices by separating different responsibilities. The transformation logic, material settings, and rendering commands all live in their own functions, making the main scene code much cleaner and easier to understand.