### Justification of Development Choices for 3D Scene

A deliberate choice was made when developing the 3D scene to include objects like cylinders, cubes, and the sphere. In making this choice, the goal was to strike a balance between satisfying the basic shape requirements and introducing a higher degree of complexity at the same time. Besides serving as a canvas for demonstrating advanced graphics programming techniques, these shapes illustrate geometric properties. Texture mapping, lighting, and shading effects can be applied and observed to complex objects like cylinders and the sphere. By doing so, the scene is transformed from a mere collection of simple shapes into an intricate and visually stimulating ensemble. This results in a more interactive learning and demonstration environment.

**User Navigation in the 3D Scene**

It has been crucial to the interactive experience to design a user-friendly navigation system within the 3D scene. This camera setup allows for a comprehensive examination of all objects and comprehensive visibility across the X, Y, and Z axes, essential for exploring the scene. Users have an intuitive and comfortable way to explore the scene with the WASD keys controlling horizontal movement and the QE keys controlling vertical navigation. With the addition of the P key, the user can toggle between various camera perspectives, including an overhead view, further enriching their experience. By providing alternative vantage points, this feature enables users to perceive objects and spatial arrangements in a different way.

**Custom Functions and Their Modularity**

To make the code more modular and organized, custom functions such as **generatePlane, generateSphere, generateSolidCylinder, generateCylinderWithHole, and generateCube** have been developed. Functions like these encapsulate specific tasks, such as generating vertices and indices. The modularity of these objects lies in their ability to be reused across multiple instances, allowing multiple shapes of varying sizes or dimensions to be created without duplicating code. By doing so, the code is not only easier to read and maintain, but it makes it easier for new features to be added or changes to be made. From setting up mesh data and loading textures to compiling shaders and handling user inputs, functions like **setupMesh, loadTexture, compileShader, createShader, processInput, and framebuffer\_size\_callback** serve distinct purposes. By organizing these functions into discrete, reusable components, the codebase is more robust, adaptable, and easy to manage because it adheres to modularity and single-responsibility principles.

**Areas for Improvement**

One area for potential improvement is the cylinder design with a hole, which stands out in relation to the design choices. There was an opportunity to further enhance its design, particularly for easier decal display, even though it serves its basic purpose. By improving texture mapping or adding extra geometric detail, a more thoughtful design could have made it easier to integrate decals seamlessly. The design elements of this specific object could have been enhanced as well. The cylinder could have been made more compelling and visually interesting if more nuanced details or unique features had been added to it instead of just being a simple geometric shape. It would not only demonstrate a higher level of graphics programming skills but would also result in a more engaging and eye-catching 3D scene as a result.