**Justification of Development Choices**

In developing my 3D scene, I aimed to create a relatable and engaging environment using everyday objects. The chosen objects: a table, a pen, Rubik's cube, coffee mug and a laptop—reflect a typical desk setup, which many users can connect with. The pen and the Rubik's cube utilize simple shapes like cylinders and boxes, keeping the polygon count low and manageable. While the mug introduce recognizable items that contribute to a realistic and detailed scene.

Programming for the required functionality involved ensuring each object met the low-polygon requirement and was textured appropriately. I used high-resolution, royalty-free images for textures to enhance realism without compromising performance. Lighting was meticulously set up to include both ambient and point light sources, with a colored light to add depth and visual interest. This approach ensured all objects were well-lit and shadows were realistic, providing a polished presentation.

**Navigation and Camera Control**

Users can navigate the 3D scene seamlessly using a combination of keyboard and mouse inputs. The WASD keys allow for forward, backward, left, and right movement, while the QE keys control vertical movement, providing full 3D spatial navigation. The mouse cursor is used to change the camera's orientation, enabling users to look around the scene by moving the mouse up, down, left, and right. The mouse scroll wheel adjusts the movement speed, offering finer control over navigation, which is particularly useful for close inspection of objects.

Setting up the virtual camera involved coding for pitch, yaw, and roll adjustments to ensure smooth and intuitive navigation. The camera's orbit radius was carefully calibrated to ensure it captures all objects in the scene from various angles without leaving any part in the dark. This setup provides an immersive experience, allowing users to explore the 3D environment thoroughly.

**Custom Functions for Modularity and Organization**

To keep my code modular and organized, I developed several custom functions. These functions handle tasks such as object creation, texture application, and lighting setup, making the code more readable and maintainable.

For example, the createObject function takes parameters for shape, size, position, and texture, and generates the specified object in the scene. This function is reusable for any new object, ensuring consistency in object creation and reducing code duplication.

Another custom function, applyLighting, sets up the scene's lighting based on predefined parameters for ambient, diffuse, and specular components. By centralizing the lighting logic, any changes to the lighting setup can be made in one place, simplifying adjustments and ensuring all objects receive consistent lighting.

The setupCamera function configures the camera's movement and orientation controls, abstracting away the complexity of navigation setup. This function is crucial for maintaining a clean main script and allows easy adjustments to camera behavior.

These functions contribute to a well-structured codebase, facilitating future enhancements and making the program more manageable. By breaking down tasks into reusable functions, I ensured that my code is not only effective but also scalable and easy to understand.