**Reflection on Development Choices for the 3D Scene**

**Choice of Objects**

The selection of objects for the 3D scene was driven by a desire to create an engaging and visually appealing environment while also considering the technical limitations and objectives of the project. The objects - a cube (representing foliage) and a plane (acting as a tree trunk) - were chosen due to their geometric simplicity and the ease with which they can be manipulated using basic 3D transformations. This simplicity allowed for a focus on other critical aspects of 3D rendering, such as texturing, lighting, and camera control.

**Programming for Required Functionality**

Programming the scene involved several key functionalities: texturing, lighting, and camera control. Texturing was implemented using OpenGL’s texture mapping capabilities, allowing the objects to have a more realistic appearance. Lighting was achieved through the implementation of the Phong shading model, which provided a realistic depiction of light interaction with the objects, enhancing the scene’s depth and realism. The addition of a second light source demonstrated the ability to create more complex lighting scenarios.

**User Navigation in the 3D Scene**

Users can navigate the 3D scene through a combination of keyboard and mouse inputs, providing an intuitive and interactive experience. The WASD keys control the forward, backward, and lateral movements, while the QE keys adjust the vertical position, offering a six-degree freedom of movement within the scene.

The virtual camera is controlled using the mouse, where moving the mouse changes the camera’s orientation, allowing the user to look around the scene seamlessly. The mouse scroll wheel adjusts the camera’s movement speed, giving the user the ability to traverse the environment more quickly or slowly, depending on their preference.

**Explanation of Custom Functions**

**Camera Class**

The Camera class is a cornerstone of the project, encapsulating all functionality related to camera positioning and orientation. It handles transformations to the view matrix based on user input, enabling dynamic scene navigation. Its methods **‘ProcessKeyboard’** and **‘ProcessMouseMovement’** allow for smooth and intuitive camera control. This class is highly reusable in any OpenGL project requiring a controllable camera.

**Shader Loading Function**

**‘LoadShaders’** is a utility function that simplifies the process of loading, compiling, and linking GLSL shaders. It takes the file paths of the vertex and fragment shader source code, compiles them, and creates a shader program. This function is modular and can be reused in any OpenGL project that requires shader compilation.

**Texture Loading Function**

**‘LoadTexture’** streamlines the process of loading image files and creating OpenGL textures from them. It is adaptable for different image formats and can be easily integrated into other projects that involve texture mapping in OpenGL.

The development choices and custom functions in this 3D scene were aimed at creating an effective learning platform for key OpenGL concepts like texturing, lighting, and camera control, while also ensuring code reusability and modularity. The choice of simple objects allowed for a focus on these techniques, and the implementation of user interaction mechanisms provided a practical understanding of 3D navigation and scene manipulation.