1. **Justify development choices for your 3D scene.**

I chose to create a fishing-themed scene with a coffee mug, tackle box, fishing rod, and a trout. Fishing is a big hobby of mine and I thought that putting these elements together in a scene would be a good display for the viewer. I specifically selected each object for the scene because they each relate to my fixing experiences. The coffee mug with the steam particles suggests an early morning atmosphere, the tackle box and fishing rod represent the main fishing equipment, the trout adds somewhat of a trophy element to the scene, and the wooden tabletop provides a sort of natural foundation that ties everything together.

1. **Explain how a user can navigate your 3D scene.**

A user can navigate my scene with keyboard and mouse functions. On the keyboard, the W key can be used to move the camera position forward, the S key can be used to move the camera position backward, the A key can be used to move the camera position left, and the D key can be used to move the camera position right. The E key can be used to move the camera position upward, and the Q key can be used to move the position downward. Also on the keyboard, the O key can be used to switch to an orthographic display and the P key can be used to switch back to the initial perspective display. On the mouse, the user can change the orientation of the camera, following the mouses movement to look up, down, left, or right. Also on the mouse, the scroll wheel can be used to adjust the camera’s movement speed. Scrolling up on the scroll wheel will increase the camera’s movement speed and scrolling down will decrease the camera’s movement speed.

1. **Explain the custom functions in your program that you are using to make your code more modular and organized.**

The code is organized with several functions that offer reusability and maintenance:

1. SetTransformations() is a core function that handles:

* Object scaling
* Rotation on all axes
* Position translation

This function is highly reusable because it encapsulates all transformation logic in one place.

1. SetShaderMaterial() and SetShaderTexture() functions manage material and texture properties, making it easy to:

* Apply consistent materials across similar objects
* Switch textures dynamically
* Maintain material properties in one location

1. LoadSceneTextures() function centralizes all texture loading, making it easy to:

* Add new textures
* Handle loading errors
* Maintain texture references

1. DefineObjectMaterials() creates a library of materials that can be:

* Reused across multiple objects
* Modified in one place
* Extended for new objects

The modular approach makes the code more maintainable and allows for easy scene modifications or extensions. For example, adding a new object only requires defining its transformations, materials, and textures using the existing functions.

The shader management system also displays good organization, with separate controls for:

* Color values
* Texture mapping
* Material properties
* Lighting calculations

This modular design allows for easy scene modifications and potential reuse in other 3D applications.