**Final Project by Valeriya Barker**

For the development of my 3D scene, I chose objects that are commonly found on a desktop setup: a monitor, a keyboard, a notebook and a cup. These objects were selected to create a relatable and realistic environment that a user can easily identify with. I also meant to add backdrop to provide context and enhance the realism of the scene by simulating a typical workspace background.

To achieve the required functionality, I implemented various methods for loading textures, setting up materials, and configuring lighting. These elements work together to render the scene accurately and ensure that each object is visually distinct and appears as intended. The textures were applied to the objects to give them a more realistic appearance, while the lighting setup ensures that the objects are illuminated in a natural and visually appealing way.

Navigation in the 3D Scene

Users can navigate the 3D scene using keyboard and mouse inputs. The keyboard controls allow the user to move the camera forward, backward, left, and right, while the mouse controls allow the user to rotate the camera and look around the scene. This combination provides a comprehensive navigation system that lets the user explore the scene from different angles and perspectives.

Virtual Camera Control Setup

The virtual camera in the 3D scene is controlled using input devices such as the keyboard and mouse. The keyboard inputs are mapped to movement functions, allowing the user to move the camera position in the scene. For instance, pressing the 'W' key moves the camera forward, 'S' moves it backward, 'A' moves it left, and 'D' moves it right. Mouse inputs are used to adjust the camera's orientation, enabling the user to look around the scene. This setup provides an intuitive and interactive way for users to navigate and explore the 3D environment.

Custom Functions for Modularity and Organization

To make the code more modular and organized, I developed several custom functions:

* SetTransformations: This function sets the transformation values (scale, rotation, position) into the transform buffer. It ensures that each object is transformed correctly before being drawn. This function is reusable for any object that needs to be transformed, making it versatile and essential for rendering different shapes.
* SetShaderColor: This function sets the color values into the shader. It is used to change the color of objects dynamically, providing flexibility in how objects are rendered. It is reusable for any object that requires a color change.
* SetShaderTexture: This function sets the texture data associated with a specific tag into the shader. It is used to apply textures to objects, ensuring that they are rendered with the correct visual appearance. This function is reusable for any object that requires texturing.
* LoadGLTextures: This function loads textures from image files and binds them to the appropriate texture units in OpenGL. It ensures that all textures are prepared and available for rendering. This function is reusable for loading and managing textures in the scene.
* DefineObjectMaterials: This function configures the material settings for objects within the 3D scene. It allows for the definition of different materials, which can be applied to various objects to achieve the desired visual effect. This function is reusable for defining and managing materials for different objects.

By organizing the code into these modular functions, I was able to create a structured and maintainable program. Each function has a specific purpose and can be reused across different parts of the program, making the code more efficient and easier to understand.