Final Project Reflection

Justification of Development Choices

For my final project, I chose to replicate my personal desk setup as a 3D scene because it provides a realistic and relatable environment. This setup includes a lavender corner desk, three monitors, and a keyboard. Selecting these objects allowed me to demonstrate a range of skills from modeling large complex shapes to adding fine details and textures. Additionally, replicating a familiar scene motivated me to achieve a high level of accuracy and detail.

Selection of Objects:

- Corner Desk: I included the lavender corner desk to add a unique touch and challenge myself with its complex shape.
- Monitors and Keyboard: These objects were selected to showcase my ability to model both large and small objects and to create a functional workspace.

Use of Basic Shapes:

- Desk: Constructed using box shapes for the surfaces and legs, and a prism for the corner piece.
- Monitors: Made with thin boxes and planes for the screens, and cylinders for the stands.
- Keyboard: A single box shape with a texture applied to represent the keys.

Texturing:

- Desk: Applied a lavender texture to accurately represent its color.
- Monitors and Keyboard: Used screen and detailed textures to simulate the display and keys.

Lighting:

- Included multiple light sources to create realistic lighting effects, such as ambient, diffuse, and specular lighting. This enhances the visual appeal and realism of the scene.

Navigation in the 3D Scene

The 3D scene can be navigated by users with standard keyboard and mouse inputs. The virtual camera is configured to rotate and move, providing a comprehensive overview of the complete desk setup. The camera controls include:

- W A S D keys: For forward, left, backward, and right movement, respectively.
- Mouse cursor: For changing the orientation of the camera to look up, down, right, and left.
- Mouse scroll: For adjusting the speed of the camera movement around the scene.
- C key: To cycle through predefined camera positions, offering different views of the scene.
- P key: Switches to a perspective (3D) view.
- O key: Switches to an orthographic (2D) view.

These controls enhance user engagement and enjoyment by allowing them to view the scene from various perspectives and distances.

To implement perspective and orthographic views, the glViewport and glOrtho functions are used. The perspective view provides a realistic 3D representation, while the orthographic view offers a flat 2D projection without perspective distortion. This is achieved by switching the function call to retrieve either the perspective or orthographic projection matrix.

- Perspective View (P key): The camera settings for perspective projection allow users to navigate the scene with a sense of depth.
- Orthographic View (O key): The camera settings for orthographic projection ensure that the camera looks directly at the 3D object, with no perspective distortion. The bottom plane should not be visible in this mode.

Custom Functions for Modular and Organized Code

To maintain modularity and organization in my code, I developed several custom functions that ensure reusability and efficiency. Here are the core functions and their purposes:

- SetTransformations()
    - Purpose: Applies scaling, rotation, and translation transformations to objects.
    - Implementation:

C++

```
void SetTransformations(glm::vec3 scaleXYZ, float XrotationDegrees, float YrotationDegrees, float ZrotationDegrees, glm::vec3 positionXYZ) {
   // Scaling
   glm::mat4 scale = glm::scale(glm::mat4(1.0f), scaleXYZ);
   // Rotation
   glm::mat4 rotationX = glm::rotate(glm::mat4(1.0f), glm::radians(XrotationDegrees), glm::vec3(1.0f, 0.0f, 0.0f));
   glm::mat4 rotationY = glm::rotate(glm::mat4(1.0f), glm::radians(YrotationDegrees),
```

```
glm::vec3(0.0f, 1.0f, 0.0f));
   glm::mat4 rotationZ = glm::rotate(glm::mat4(1.0f), glm::radians(ZrotationDegrees),
glm::vec3(0.0f, 0.0f, 1.0f));
   glm::mat4 rotation = rotationZ * rotationY * rotationX;
   // Translation
   glm::mat4 translation = glm::translate(glm::mat4(1.0f), positionXYZ);
   // Combine transformations
   glm::mat4 model = translation * rotation * scale;
   // Apply to shader
   glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
}
```

        - Reusability: This function can be used for any object in the scene, ensuring consistent and efficient application of transformations.
- SetShaderColor()
    - Purpose: Sets the color values into the shader for rendering objects without textures.
    - Implementation:

C++

```
void SetShaderColor(float redColorValue, float greenColorValue, float blueColorValue, float
alphaValue) {
   glUniform4f(glGetUniformLocation(shaderProgram, "ourColor"), redColorValue,
greenColorValue, blueColorValue, alphaValue);
}
```

        - Reusability: Allows for quick color changes to any object, making it versatile for various parts of the scene.
- SetShaderTexture()
    - Purpose: Binds a texture to an object for rendering.
    - Implementation:

C++

```
void SetShaderTexture(std::string textureTag) {
   glBindTexture(GL_TEXTURE_2D, textureMap[textureTag]);
}
```

        - Reusability: Can be used to apply textures to various objects, enabling quick texture changes and consistent application across the scene.
- LoadSceneTextures()

    - Purpose: Loads and binds all necessary textures for the scene.
    - Implementation:

C++

```
void LoadSceneTextures() {
   textureMap["desk"] = LoadTexture("textures/Desk_texture.jpg");
   textureMap["monitor"] = LoadTexture("textures/Monitor-Screen_texture.jpg");
   textureMap["keyboard"] = LoadTexture("textures/Keyboard_texture.jpg");
   // Additional textures...
}
```

    - Reusability: Streamlines the texture loading process, ensuring that all textures are prepared before rendering.
- SetupSceneLights()
    - Purpose: Configures and positions the lights in the scene to achieve the desired lighting effects.
    - Implementation:

C++

```
void SetupSceneLights() {
   // Set light properties
   glUniform3f(glGetUniformLocation(shaderProgram, "lightPos"), lightPos.x, lightPos.y, lightPos.z);
   glUniform3f(glGetUniformLocation(shaderProgram, "viewPos"), cameraPos.x, cameraPos.y, cameraPos.z);
   glUniform3f(glGetUniformLocation(shaderProgram, "lightColor"), 1.0f, 1.0f, 1.0f);
   glUniform3f(glGetUniformLocation(shaderProgram, "objectColor"), 1.0f, 0.5f, 0.31f);
   // Additional light configurations...
}
```

    - Reusability: Allows for easy adjustments to the lighting setup, making it adaptable to different scenes.

Milestone Progress

1. Milestone One: Project Proposal
    - Proposed to recreate my personal desk setup in 3D, including a lavender corner desk, three monitors, and a keyboard. The proposal outlined the basic shapes needed and the textures to be applied.
2. Milestone Two: Beginning a 3D Scene

    - Began transforming the basic shapes for the desk and monitors, modeling the primary surfaces and ensuring accurate positioning to reflect the real-world setup.
3. Milestone Three: Interactivity in a 3D Scene
    - Incorporated input devices and camera movement, allowing users to navigate the 3D scene, adding depth to the project and making it more engaging.
4. Milestone Four: Texturing Objects in a 3D Scene
    - Applied textures to the desk, monitors, and keyboard. This step was crucial for achieving a realistic look as the textures added detail and depth to the objects.
5. Milestone Five: Lighting Complex Objects
    - Applied lighting to the scene, including ambient, diffuse, and specular lighting, enhancing the visual appeal and creating realistic shadows and highlights. Also LED lighting under the upper desk surface was added to replicate the look from the reference image.

Final Project Submission

The final project submission brings together all the elements from previous milestones, refined based on feedback. The submission includes the complete 3D scene and a document explaining the design decisions. This project demonstrates my ability to model, texture, and light a realistic 3D scene, highlighting the skills learned throughout the course.

This reflection covers the development choices, navigation setup, and custom functions used in the project, aligning with the requirements of Milestone Five and the final project submission.