**Reflection on 3D Graphics Project**

**9. Justify development choices for your 3D scene. Think about why you chose your selected objects. Also consider how you were able to program for the required functionality.**

When embarking on this project, I had to make several development choices that shaped the final outcome of my 3D scene. The overall concept I decided to model was a simple mug placed on a table. This might seem like a relatively straightforward scene, but the decision was motivated by the challenges I was encountering, especially as I navigated through the complexity of OpenGL, shaders, and textures. By selecting a more simplified object, I was able to focus on the fundamental aspects of the project while still meeting the basic requirements.

My struggles with the project started early, particularly with the organization of the various files. Initially, I had difficulty understanding how the .glsl shader files interacted with my code. The ShaderManager.h and ShaderManager.cpp files were, somewhat confusingly, located in the textures folder, which I only realized later on in the development process. This oversight led to some delays as I was piecing together how the shaders were supposed to be managed in the context of the scene I was building.

Another significant challenge I faced was with textures and lighting. I was able to grasp most of the initial setup, such as loading basic shapes and creating a 3D scene, but once it came to applying textures and integrating proper lighting, I found myself struggling. Although I was eventually able to get the lighting code to run, I suspect that the linker errors I encountered were related to naming conflicts between variables or functions, potentially due to unintentional overlaps with other parts of the project. This issue, combined with the stress of running into nearly 250 warnings, made the process difficult. Despite these challenges, I managed to simplify the scene to focus on getting the core logic to work correctly, allowing me to successfully render the mug on the table.

**10. Explain how a user can navigate your 3D scene. Explain how you set up to control the virtual camera for your 3D scene using different input devices.**

In terms of navigation, I wanted the user to have an intuitive and straightforward way to explore the 3D scene. I implemented a simple camera control system using familiar keybindings. The AWSD keys allow the user to move the camera left, right, forward, and backward, while the QE keys control vertical movement (up and down). Additionally, I added a click-and-drag functionality that lets the user rotate the camera by holding down the mouse button and dragging the mouse. These controls were chosen to mirror the style of navigation seen in many 3D modeling or gaming environments, making it easy for the user to navigate through the scene and view the objects from different angles. The primary goal here was to offer a balance between simplicity and functionality, ensuring that the navigation experience was intuitive without overwhelming the user with unnecessary complexity.

**11. Explain the custom functions in your program that you are using to make your code more modular and organized. Ask yourself, what does the function you developed do and how is it reusable?**

To keep the code organized and as modular as possible, I broke down the functionality into multiple header (.h) and source (.cpp) files. Each header file defines specific functions or variables that are then implemented in the corresponding source files. This structure allows the code to remain more manageable, even though I encountered issues with organization. It also makes it easier to debug, as different sections of code are compartmentalized. Although my project had several issues, I made sure to document my thought process by adding comments throughout the code. This would help any future developers (or myself) to follow the logic and reasoning behind certain decisions or functions.

One of the custom functions I created was aimed at handling camera movement. By placing the camera control logic into a separate function, I could reuse this setup in different parts of the project without duplicating code. This approach to modularity not only reduces redundancy but also makes it easier to test specific aspects of the code without affecting the entire project. Another example was the object rendering code, which was separated into its own function, allowing for a cleaner and more readable main loop.

As I approach the submission of this project, I want to acknowledge that, despite my best efforts, there are still some bugs and issues that I haven't been able to resolve. The development process has been quite stressful for me, especially with the numerous warnings and linker errors that caused confusion and anxiety throughout the project. While I've done my best to simplify and organize the code, there are aspects—particularly with textures, shaders, and overall project structure—that I still find difficult to grasp.

Given the challenges I've faced, I would really appreciate any additional support or guidance on how to make the project fully functional. If possible, it would be extremely helpful if you could provide a zip file or some form of detailed instructions to help clarify what needs to be fixed or improved. I'm eager to learn and improve my understanding, but I feel that I need some extra help to fully grasp the areas where I got stuck.

Thank you for your time and consideration. I look forward to any feedback you may have that can help me refine my work.