Final Project

Raymond S. Cooke

Southern New Hampshire University

Reflection

When exploring objects for this project, I selected simple shapes that make a scene comprised of a stick of butter, marble, and can of Pringles sitting atop a plane. As the course progressed, I struggled with implementing multiple objects. As new aspects of three-dimensional object programming were introduced, I was unable to comprehend how to organize them in relation to more than one object. I was not able to program for the required functionality.

To create the cylinder and sphere, pre-made classes were used to generate the vertex attribute and vertex buffer objects. As texture and shaders were added, these objects became extremely distorted. For the stick of butter and plane, I was able to render consistently but failed to apply texture and shading. If I could take this course again, I would focus more on multiple objects and how to initialize them, their pointers, and characteristics.

A user can navigate the scene by using the “W” key to move the camera forward, the “S” key to move the camera backward, the “D” key to move the camera right, the “A” key to move the camera left, the “Q” key to lower the camera, and the “E” key to lower the camera. A user can use the “P” key to toggle between a perspective and orthographic view of the scene. This feature enables the viewer to view the scene as two-dimensional. Directional mouse inputs enable the viewer to adjust the camera’s gaze and the mouses scroll wheel enables users to adjust the sensitivity of all inputs.

After creating the code to switch views, the response to user input was extremely sensitive. A custom function was implemented within the “URender” and “UProcessInput” methods that slowed down the response of the switch between perspective and orthographic views. After applying this code, the view ceased to change every time the render loop was iterated. This was accomplished by declaring a 0.6 second period before the “P” key would be able to be used more than once. This waiting period yielded a speedy yet reasonable response to the user defined view.

The program was written in one source file but divided into functions. Each object and its meshes were divided into its own unique data structure that stored its VAO, VBO, and indices. Shader programs were defined before the main function where all applicable functions were called. This includes window creation, shaders, mesh creators/destroyers, texture invocations, and the render loop. This approach enables a developer to copy and reuse pieces of the source code. The custom functions were not exactly intended to make the code more modular, but to add functionality to the program.

References

Last Name, F. M. (Year). Article Title. *Journal Title*, Pages From - To.

Last Name, F. M. (Year). *Book Title.* City Name: Publisher Name