**7-1 Final Project Submission:**

Design Decisions Document

**Reflection:**

This document is both a guide to my program and the decisions I made during the development process. To start with the image I selected, I chose my image based on personal taste, however I had not accounted for the full requirements of the final project. This led me to taking some artistic liberty in my recreation. Starting with the seemingly random pyramid in my scene. The final submission required that I include at least four different primitive geometries, most of my image could be recreated with only three, thus I added the pyramid. I chose objects that would still convey a similar artistic feeling as the reference image.

Moving on to lighting choices, I decided to implement two light sources. One was assigned a color of white to simulate the general type of light we are all familiar with and that would not distort my textures. The second light was assigned a color of black to help simulate shadows. Additionally, I chose to implement moving light sources to add movement to my 3D scene as well as to demonstrate my shader programs capabilities in reflecting light.

Textures, originally, I was only going to texture two objects, however upon reflection this made objects feel out of place and I made the decision to texture all objects. I made some design changes from my reference image as the scene was being overpowered by the map image I had selected as my base plane’s texture. Instead, I went with a nice dark wood texture that would allow for my objects to stand apart from the base plane. I textured the pyramid in the scene after the pyramid I had been working with on the weekly assignments to add a layer of my experience throughout the course. The rest of the textures were chosen to enable an easier recreation of my reference picture, while still conveying an objects composition.

**Scene Navigation:**

To navigate my scene I decided to stick with common control schemes for todays standards. Below is a list of controls and their functions:

Keyboard Controls:

|  |  |  |
| --- | --- | --- |
| Key | Action | Description |
| W | Move Forward | Move the camera forward |
| S | Move Backward | Move the camera backward |
| A | Move Left | Move the camera left |
| D | Move Right | Move the camera right |
| E | Move Up | Move the camera up |

|  |  |  |
| --- | --- | --- |
| Key | Action | Description |
| Q | Move Down | Move the camera down |
| 1 | Change View  Perspective | Change the view to a perspective frame (3D) |
| 2 | Change View  Orthographic | Change the view to a orthographic frame (2D) |
| Arrow Up | Move Light Up | Move the white light cube in the positive Y direction |
| Arrow Down | Move Light Down | Move the white light cube in the negative Y direction |
| Arrow Left | Move Light Left | Move the white light cube in the negative X direction |
| Arrow Right | Move Light Right | Move the white light cube in the positive X direction |
| Shift + Arrow Left | Move Light Away | Move the white light cube in the negative Z direction |
| Shift + Arrow Right | Move Light Closer | Move the white light cube in the positive Z direction |

**Functions:**

I had two primary goals when designing functions within my program: A) Practice consistency by matching the given functions and B) Reusability. If I am to create a function, there needs to be some form of consistency, either to what was given to me and/ or within my own style. Additionally, if I am going to create a function I want to ensure that it can be used in a variety of ways that enhance the programs capabilities. In focusing on conformity, I followed the naming conventions of the functions laid out for us and continued it through my own functions. (UInitialize, UCreate…, UProcessInput, etc.) Also I tried to create segmented code to allow my main function to remain relatively small while still conveying my program’s intent. ULoad functions were designed to either load batches of data or segment repetitive calls of similar types. For example, ULoadtextures handles the multiple calls to the UCreateTexture function and allows for easy updating to add new textures to the program. UCreateMeshObjects does a similar thing, but pertaining to my shape object creation calls. UCreatePyramid, Plane, Cube, and Cylinder functions hold the data and vao (Vertex Array Object), vbo (Vertex Buffer Object), and/or ibo (Index Buffer Object) as well as the attribute pointers for each mesh. If given more time, I would extract the functions into their own headers and cpp files to continue compartmentalizing my program. I only focused on one Class object outside of the functions due to the complexity of creating a cylinder mesh. (Cylinder.h and Cylinder.cpp) In the creation of this cylinder class I adapted code from Song Ho Ahn’s cylinder tutorial to match the needed functionality of my cylinder class. (Ahn, n.d.)

References

Ahn, S. H. (n.d.). OpenGL cylinder, Prism & Pipe. Retrieved December 15, 2022, from http://www.songho.ca/opengl/gl\_cylinder.html