Dan Stull

CS330

12/05/2022

Reflection

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

For this project, I decided to take a picture of my computer desk (and the objects on it). I chose this area because it was somewhere I often sat, if not multiple times a day, for many hours. It was an area I knew well and could picture without much thought. The objects represented in the photo were objects that were, for the most part, always there. So there wouldn’t be any confusion when it came to time create the final scene. And as such, this allowed me to visualize the objects better in order to program them. The cups (mug and plastic cup) were similar shapes, so I knew I could reuse some code to satisfy the requirements for those two. The mouse was a simple enough shape, one that could be satisfied with a few of the simple shapes. And the desk and keyboard were, like the cups, similar in shape, which meant I could reuse code.

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

The user of the program can maneuver the scene by using the WASD keys for forward (W), backward(S), left (A), and finally, right with (D). They can also “lift” and “lower” the scene using (E) and (Q), respectively. This allows for multiple views using the keyboard and nothing else. I was able to give this maneuverability by utilizing the WHILE loop in the main function. As the program gets initiated, a loop is run. And as long as the loop (WHILE LOOP) is still being satisfied, which is to say that the program hasn’t ended, everything within the while loop gets run over and over. With this knowledge, I used GLFW functions that work directly with the view camera. These functions, in essence, take the input, say W, for forward. Forward would be moving in the positive Y direction. The function then uses the input of 1 and moves the camera forward 1 unit. This all happens on one pass of the while loop. On the next pass, if the user is still pressing the W for forward, they are moved again. This works for each set key; it is how the user can move without lifting and pressing (repeatedly) the assigned keys to move in any direction. The use of the mouse works very similarly. As the user pans the mouse left or right (X axis) or up and down (Y axis), positive and negative inputs, from a range of 0 to 1, are used to “adjust” the camera accordingly. While using the mouse, the user can also move forward and backward with the mouse’s spin wheel. Also, to move faster in the Y direction (forward and backward), the user can simultaneously use the W key and scroll forward on the wheel, or, to move backward faster, the user can press S and scroll backward, taking advantage of both inputs simultaneously.

**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?

I knew I would be drawing quite a few objects to the screen. And because of this, I knew I didn’t want to clutter the draw function up with a TON of code. To remedy this, I ensured each object had its own function. I took care of the texturing, translating, scaling, and rotation in that function. And in the Draw() function, I called each object’s function. In this way, the Draw function is re-used many times, once for each object drawn to the scene. This results in less redundant code and less clutter.

Also, knowing I was going to use the WHILE LOOP in the main function to move the camera, I didn’t want to clutter the main function, so I created ProcessInput() function inside the loop. And each time the loop runs it calls the function from outside the main.

Another example of modular coding is the CreateVertices() function that is called in the main, just above the while loop. This function needs only to be called once, and in doing so, I can create all the vertices, VBO’s and VAO’s in one go. Had I not created this function, I would have had to call each object’s vertices one at a time, leading to a very cluttered main function.

Lastly, I took advantage of function declaration to create an “easier” to read program. This means I declared each function in the program before the main and then defined each after. Doing this doesn’t make the program run better but it makes it a little easier to read. The variables and function declarations are first, and then the main function. And lastly, the definitions for the functions. I found this more aesthetically pleasing.