**Design Decisions**

David Smith

Department of Computer Science, Southern New Hampshire University

CS 330: [Computational Graphics and Visualization](https://learn.snhu.edu/d2l/home/794495)

Professor E. Rodriguez

August 15, 2021

For this project, each object from the original image except the laptop was represented with as few simple 3D shapes as possible. The couch is made up of many cuboids and two cylinders. The end table is composed of cylinders and cuboids as those are the shapes in the original object. Likewise, the coffee table contains several cuboids for legs, top surface, and underpinning. The walls, floor, and fluorescent light are made from planes, and the trim for the wall is another pair of cuboids. The lamp is made up of a hemisphere, a cylinder each for the stand and shade, and a sphere for the light bulb.

While creating this project, there were several times when creating an object required repetitive calls to certain library functions. To reduce these, custom functions like CreateVAOS and TestResource were created. These functions made it easier to use loops to create VAOs for meshes and to test shader compilation. Both functions would be useful in creating other OpenGL applications.

Creating a plane was a fairly simple matter without a lot of code required, so I chose to leave that functionality in a function. Cuboids, spheres, and cylinders required more effort and code to represent, so I chose to create a class for each. Each of these classes have a GetVertices and GetIndices method to obtain data necessary for building the VAOs and VBOs. Because these classes programmatically build the shapes, they were instrumental in creating the different pieces to build each complex object. Each of these classes would also be helpful in creating future OpenGL applications.

Several of the objects in the scene use more than one shape. A mesh struct was created to hold the data needed to pass into the CreateVAOS function. This made it easier to loop through a vector of meshes from different shape classes to build out a completed object using a loop. Each struct also contains information on which texture to use for the object.

The coffee table, end table, couch, and lamp each have their own function for building the objects. These functions could be reused in other projects along with the shape classes and CreateVAOS function to recreate similar objects. The adjustments made to scale, rotation, and transpose in the PlaceObjects function could be altered to place these objects in different positions to create a different room layout. Each mesh has a texture variable that can be adjusted to make the objects look different while still maintaining the same shape.

One idea for future modifications to make this code more modular would be to place the translate and rotate matrices for each mesh as a variable in the mesh. This would allow for looping of placing the meshes in the scene. Some other useful simplifications would be to place all of the light information into structs, load the shader code from files, and creating a configuration file for loading the textures.

The lighting used includes a fluorescent light fixture that would be on the ceiling in the scene if there was a ceiling as well as a lamp on the end table. I chose these locations as these are two of the lights in the actual room that the original picture represents. They are in locations that make for good lighting of the scene. The light colors were chosen to represent close to the actual light coloring in the room. However, to get a feel for how the light interacts with the environment, I added a feature to change the colors of the lights in the scene. Pressing F1 will toggle the lamp color between the default and an orangish color. Pressing F2 will toggle the fluorescent light between the default and a green color.

Other controls include Q and E which will move the camera position up and down respectively. Pressing A or D strafes the camera left or right, and pressing W or S moves the camera forward or backward. The application defaults to perspective projection, but it can be toggled to orthographic and back with a press of the P key.

Moving the mouse causes the camera to turn in the direction of movement. Scrolling the mouse wheel up or down increases or decreases the camera direction and movement speed using both the keyboard and the mouse. One of the more useful features was adding the ability to change the speed of the camera movement. After adding this, it was much easier to move around the scene and get near objects quickly while still being able to slow down movement to get closer to the objects without accidentally going through them. This was a tremendous help for making sure that the tables and couch were firmly placed on the floor and that the cushions had a little space in between them.