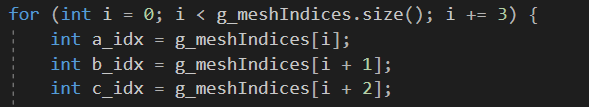
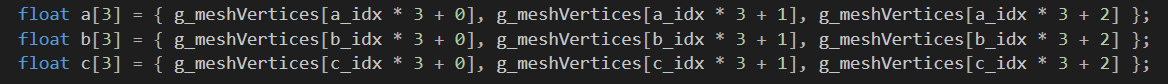
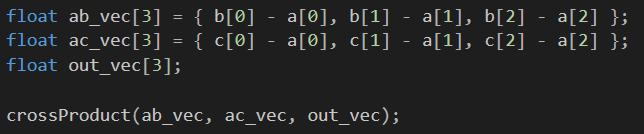
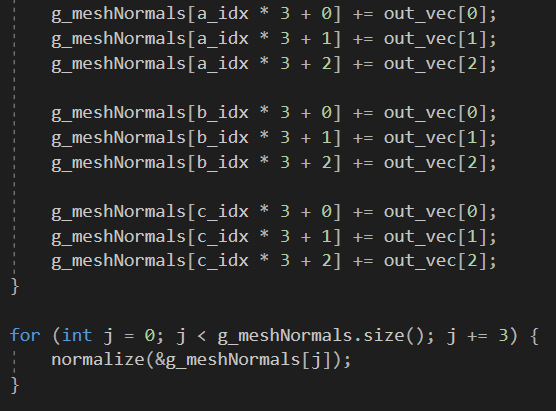
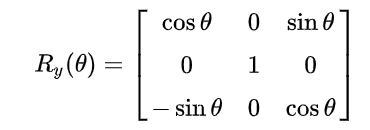
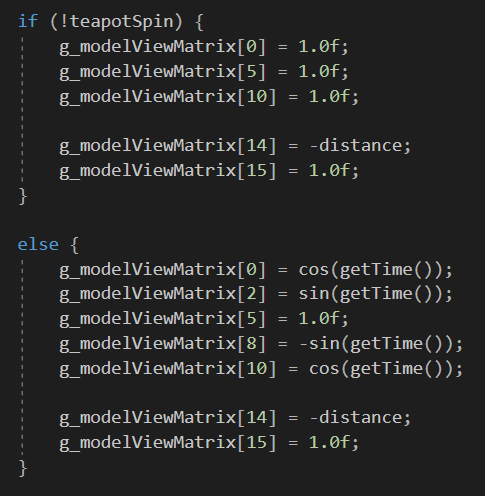
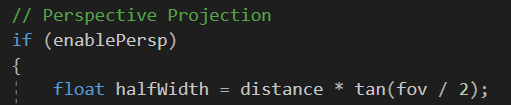
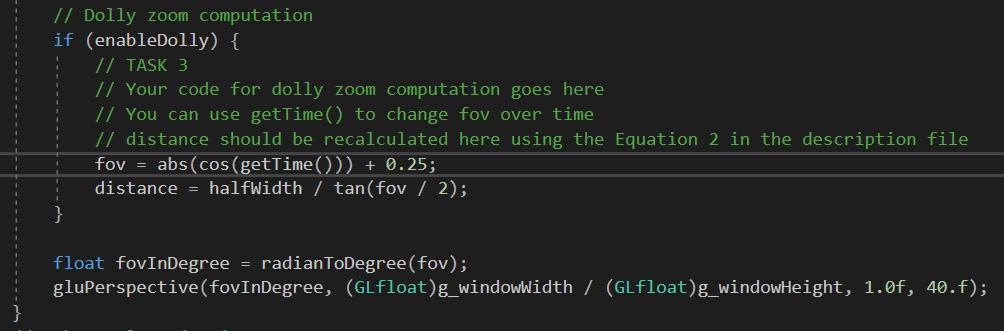
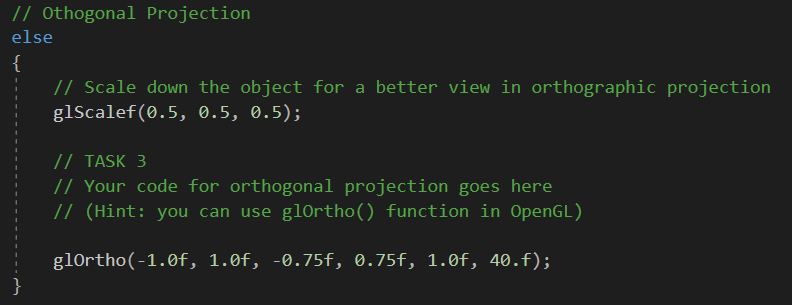
**Part 1**

* For starters, to find the normals for each vertex, we need to iterate through all triangles in the mesh. I do this by getting three values at a time from g\_meshIndices, since all three points on every triangle are next to each other in the g\_meshIndices vector. I call the three vertex indices a\_idx, b\_idx, and c\_idx. 
* I then find the x, y, and z components of these vertices using the index and multiplying by three to traverse g\_meshVertices. 
* Next, I find the normal value for the triangle by finding vector AB, vector AC, and taking the cross product, storing it in out\_vec. 
* Finally, I go to each vertex and add the normal vector to it. To average them, I then normalize each vertex in g\_meshNormals in a separate for loop. 

**Part 2**

* In order to rotate the teapot about the y axis, I just had to use the rotation matrix for three dimensions (no adjustments needed to be made since the object is at the origin.) I found a reference to this on <https://en.wikipedia.org/wiki/Rotation_matrix>. 
* Given the default settings, it seemed that g\_modelViewMatrix was a 4x4 matrix, and I assumed that the transformation matrix for 3d was found in the upper left. After testing, I found that this was the case. So using the matrix above (where the top row is has indices 0, 1, 2, the middle row has 4, 5, 6, and the third row has 8, 9, and 10), I just changed the values in the corresponding indices, using getTime() for theta to have it change rotation over time. I also put in a conditional to have it only spin after pressing the “s” key. 

**Part 3**

* As per the instructions, I first computed halfWidth using equation 1. 
* I then change fov over time using cos(getTime()), similar to part 2. I take the absolute value of that to ensure I only get positive values and assign it to fov. Since this can occasionally give me values close to zero (causing distance to become greater than zFar), I add 0.25 to ensure distance will stay within the bounds. Finally, I compute distance using equation 2. 
* For the orthographic projection, I used the hints provided and just used glOrtho. After testing with left = -1.0f, right = 1.0f, bottom = -1.0f, and top = 1.0f, I found that gave a squashed result. Seeing that the window width was 800 and the height was 600, I figured I need to make the bottom and top 0.75 times the left and right values. After doing so, the proportions of the teapot were corrected. 

Youtube link: <https://youtu.be/h4Znrfhc5ms>