

**Project 2: JOGL 3D Graphics**

**Zachary Finnegan**

**CMSC 405 Computer Graphics**

**4/12/20**

**Test Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Output | Actual Output | Pass? |
| The ‘Z’ and the ‘F’ | The ‘Z’ should look like a ‘Z’ and the ‘F’ should look like an ‘F’ with the light source coming from the upper left-hand front corner of the world. Only in the beginning though as there is not actual light. The ‘shadows’ are coded in. The ‘Z’ and the ‘F’ should be next to each other and move together while inside of the frames. They will rotate in the y and z axis and shrink to almost nothing and return to normal size repeatedly.  The ‘ZF’ rotate counterclockwise from the perspective of the viewer around the z axis and then rotate clockwise around the y axis. This results in a weird roll.  It is hard to tell how it is moving unless only one rotation is applied at a time.  All objects will appear in a 640, 480 panel |  |  |
| The Inner Frame | The inner frame frames the ‘ZF’ which weirdly happen to be my initials. This rotates in a forward roll along the x axis only. Which can be seen in the screenshots to the left.  The frame is shadowed appropriately in the starting position. This was done to 1) show I could do it 2) show I thought to do it and 3) to help show that the objects were 3D. |  | Pass |
| The Middle Frame | The middle frame frames the inner frame and is surrounded by the outer frame. The middle frame is created by increasing the size of the inner frame object. This frame rotates clockwise around the y axis.  This roll can sort of be seen in the above screenshots. Obviously, motion is hard to show in photos. |  |  |
| The Outer Frame | The outer frame frames the middle frame. It rotates counterclockwise around the z axis and is translated back and forth along the z axis. To the viewer this appears that the frame is getting closer and farther away. There is object clipping during this move. | Close to the user and in front of the other objects    Far from the user and behind the other objects |  |
| The Ball | I decided to add a ball as I was finishing this document. I wanted show I could add one and that I had 6 objects. Originally, I thought creating 5 complex objects made from lots of sub-objects would be enough, but I wanted to make sure so here we are.  The ball is multicolored as I change the color as it is being drawn. It goes from blue to purple to red. It orbits at a ~45-degree angle around all the objects. |  |  |

**Lessons Learned**

So, this was hard. Creating 3D objects is no joke. The ‘Z’ is made up of 4 unique objects that need to be properly positioned and colored. The ‘F’ uses a lot of scaling to get the rectangle objects to be the appropriate sizes in addition to translation and rotation. Getting the base objects into the right positions was difficult.

It got a little easier when I realized all translations were happening based on the original object’s location. When I realized that I started making more custom objects with better origin points and rotations. It was just easier than trying to find the correct rotation and translation of a previously used object.

For the Z and the F, I used the Matrix Push/Pop method that is used in the book and the UnlitCube.java file. I found this to be a pain and confusing. For the frames I used the Matrix Push/Pop method only for large chunks of the frame object instead of every piece. For instance, I made the front face, the inner and outer edges in one method each by creating rectangles in the correct spots. I found this A LOT easier than having standard pieces of the objects that I then positioned later.

The thing I learned that I am most proud of is my solution to the problem of changing the color of a repeated object with the appropriate shading. This could have easily gotten out of hand. My solution was to have the r, g, b parameters multiplied by the shade I wanted that part of the frame to be. This way all I had to do was pass a 1 into the color parameter I wanted and 0 into the others so I would get the right shade and color without repeating a lot of work.

This would be harder for off colors or colors that aren’t red, blue and green, but I think I could have figured something out.

I rejected the animation solution found in the JoglStarter.java as it felt complicated and I didn’t want to have to figure it out. I initial used the UnlitCube.java movement to view my objects from different angles. I felt it would be a simple matter to translate that into something automated. So, I grabbed the FPSAnimator I learned about at tutorialspoint.com and used that to control the calling of the display method. I then used global variables as increment values and the values to be incremented at the bottom of the display method to be incremented with each call and those incremented values to update the rotation, translation and size of the objects.

The sphere I grabbed from a random person on the internet and tweaked it a little as the original code didn’t use the radius parameter… I learned this was easier than figuring out the math to make a sphere with this software. I tried using the GLU interface to make a sphere at first, but I couldn’t figure out how to make it more than one flat color and thus appear like a circle and not a sphere.

Fun project.

**References**

Eck, D. J. (2018). *Introduction to computer graphics.* Hobart and William Smith Colleges.

Eck, D. J. (2018). *JoglStarter.java.* <http://math.hws.edu/eck/cs424/graphicsbook2018/source/jogl/JoglStarter.java>

Eck, D. J. (2018). *UnlitCube.java.* <http://math.hws.edu/eck/cs424/graphicsbook2018/source/glut/unlit-cube.c>

Mostafizar. (2014, October 20). *3D Sphere OpenGL.* Stack Overflow <https://stackoverflow.com/questions/5799609/3d-sphere-opengl>

Parker, S. (2017, February 23) *Java opengl 2d game tutorial – Episode 2 – Gl event listener.* YouTube. <https://youtu.be/j1DvKAGXujg>

TutorialsPoint: Simply Easy Learning. (n.d.). *Jogl – Quick guide.* TutorialsPoint. <https://www.tutorialspoint.com/jogl/jogl_quick_guide.htm>