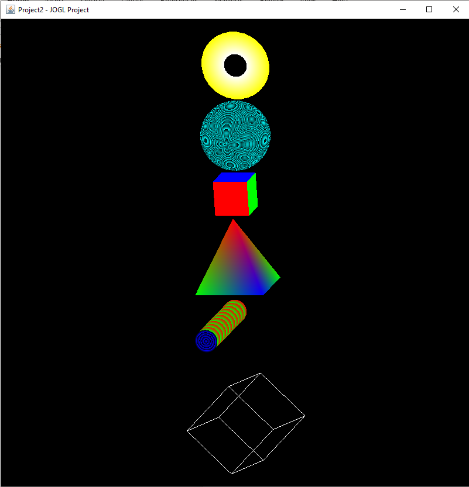
Alexander Szelestey

February 7, 2020

Project 2 – JOGL OpenGL Project

CMSC 405 6380

**Overview**

This project is a construction of a ­­Java OpenGL graphics scene with 6 unique shapes, 16 different transformation methods, and a size of 800x800.

**Users Guide**

To run this project, it will be easier to utilize a Java Integrated Development Environment (IDE) like Eclipse and you will need a full keyboard to utilize all transformation methods. There is only one class MyShapes.java. Run MyShapes class and you will be prompted with a new GUI window “Project 2 – JOGL Project” (Figure 1.). The graphics are controlled by keystrokes:

Figure 1: Main

* Left and Right Arrow Keys Rotate Image on its Y-Axis.
* Up and Down Arrow Keys Rotate Image on its X-Axis.
* Page Up and Page Down Keys Rotate Image on its Z-Axis.
* Home Key Makes all the Axis Equal to 0.
* Plus and Minus Keys Make the Triangle Bigger or Smaller.
* W and S Letter Keys Move the Rectangle Up and Down.
* A and D Letter Keys Move the Rectangle Left and Right

These instructions are printed to the console for users that need assistance navigating the window.

**Features**

This program features the use of six custom made images that transform around a GUI by the users’ keystrokes. I began this project with utilizing JoglStarter class that was given to us in the jogl.zip file. I also utilized UnlitCube and TexturedShapes classes to help with the implementation and creation of certain shapes. I utilized different methods to make a box, I used GL\_QUADS vertices to make a filled in cube. Whereas the rectangle I created out of GL\_LINES. The rectangle is not colored black, it is actually just white lines constructed to look like a 3d bordered/skeleton rectangle. I had assistance from an online forum Tutorials Point to create this image (JOGL 3D Graphics). Other features consist of translating these images around the screen to be organized in a straight line from top to bottom. Also, custom scaling so al the images fit on the canvas nicely. I also included more transformation methods to improve my knowledge and it was fun to implement. Surprisingly implementing the plus sign keystroke was the hardest one. Because I had to link it with the equals sign, or it wouldn’t work and I also had to change my scaling qualities so the triangle would get bigger and smaller.

**Test Case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** | **Figure** |
| 1 | Left Arrow Key | All images rotate left 15 on the Y-Axis | All images rotate left 15 on the Y-Axis | Pass | 2 |
| 2 | Right Arrow Key | All images rotate right 15 on the Y-Axis | All images rotate right 15 on the Y-Axis | Pass | 3 |
| 3 | Down Arrow Key | All images rotate down 15 on the X-Axis | All images rotate down 15 on the X-Axis | Pass | 4 |
| 4 | Up Arrow Key | All images rotate up 15 on the X-Axis | All images rotate up 15 on the X-Axis | Pass | 5 |
| 5 | Page Up Key | All images rotate up 15 on the Z-Axis | All images rotate up 15 on the Z-Axis | Pass | 6 |
| 6 | Page Down Key | All images rotate down 15 on the Z-Axis | All images rotate down 15 on the Z-Axis | Pass | 6 |
| 7 | Plus/Equal Key | Triangle gets scaled bigger by 0.01 | Triangle gets scaled bigger by 0.01 | Pass | 7 |
| 8 | Minus Key | Triangle gets scaled smaller by 0.01 | Triangle gets scaled smaller by 0.01 | Pass | 8 |
| 9 | “A” Key | Rectangle translates left by 0.25 | Rectangle translates left by 0.25 | Pass | 9 |
| 10 | “D” Key | Rectangle translates right by 0.25 | Rectangle translates right by 0.25 | Pass | 9 |
| 11 | “W” Key | Rectangle translates up by 0.25 | Rectangle translates up by 0.25 | Pass | 9 |
| 12 | “S” Key | Rectangle translates down by 0.25 | Rectangle translates down by 0.25 | Pass | 9 |
| 13 | Home Key | All images x, y, and z axis = 0. | All images x, y, and z axis = 0. | Pass | 10 |

**Screenshots**

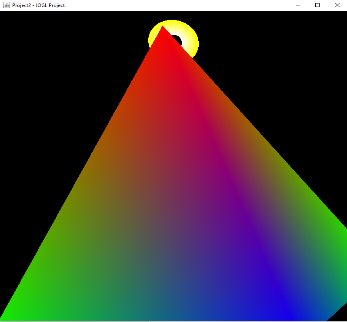
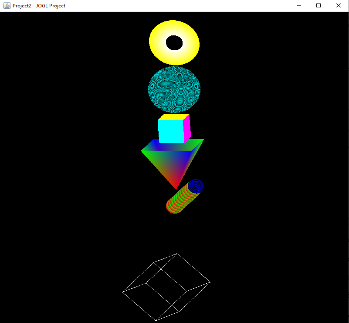
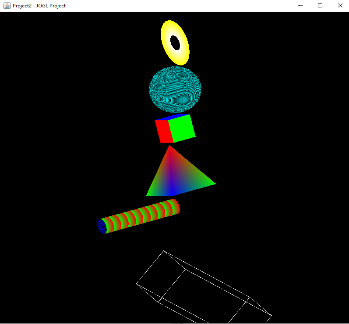
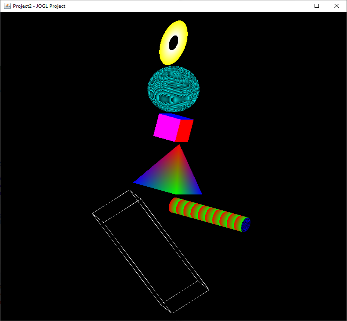


Figure 2: Left

Figure 3: Right



Figure 5: Up

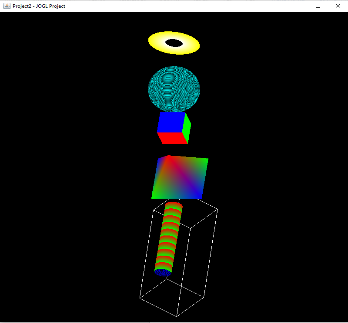


Figure 4: Down

Figure 6: Page Up/ Page Down

Figure 7: Plus +

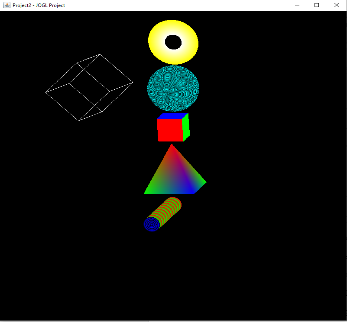
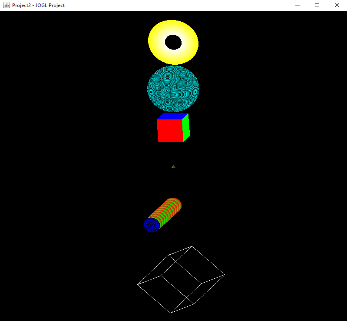


Figure 8: Minus -

Figure 9: ASWD

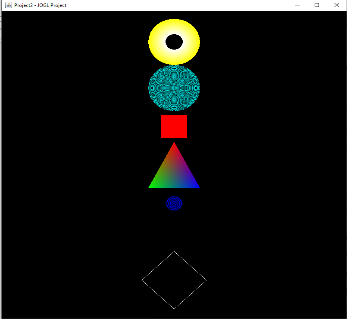


Figure 10: Home

**Lessons Learned**

The original intention was to have a skewer of shapes. I drew my rough draft on a piece of paper that had a platform with a cylinder on top of it. Then I had all the images intersecting the cylinder. However, I started creating the square, triangle, oval and once I got to the cylinder it started dancing all over the screen and I thought well that won’t work. But as I’m typing this, I just realized I should have just made it a fixed image on the screen by deleting the rotation commands. I can’t believe something so simple never clicked in my brain until now. I do wish I could have implemented a texture of some sort. When I was making the sphere into a globe I was getting a lot of errors and I figured maybe it would be easier to use two classes to implement; but that never worked so ill have to give it a shot another time. There are plenty more resources I used to complete this project but did not include them. The two main places I go for help is stackoverflow and tutorialspoint. Lastly, there are errors that pop up on my screen, but it is in regard to me not running an older version of java; doesn’t affect the code to compile (Figure 11).

**A screenshot of a cell phone

Description automatically generated**

Figure 11: Nontoxic Errors

References

Draw Pixel Art Online. (n.d.). Retrieved from <https://www.pixilart.com/draw>

Jogl.zip (2017, March 27) Retrieved from <https://www.umgc.edu>

UnlitCube (2017, April 7) Retrieved from <https://www.umgc.edu>

Henrik 1. (1961, April 01). 3D sphere opengl. Retrieved February 08, 2020, from <https://stackoverflow.com/questions/5799609/3d-sphere-opengl>

JOGL 3D Graphics. (n.d.). Retrieved February 08, 2020, from <https://www.tutorialspoint.com/jogl/jogl_3d_graphics.htm>