

**JOGL OpenGL – Project 2**

Vincent Testagrossa

CMSC 405 6380 Computer Graphics (2232)

Professor: Lauren King

Due February 7th, 2023 11:59 PM

Contents

[Project Description 3](#_Toc126684723)

[Program Usage 3](#_Toc126684724)

[Testing 4](#_Toc126684725)

[Figures 6](#_Toc126684726)

[Lessons Learned 7](#_Toc126684727)

# Project Description

The purpose of Project 2 was to use Java openGL to draw a 3D scene. The scene needed to contain at least 6 shapes and 6 different transformation methods.

# Program Usage

Once the scene is loaded, the user may use the ‘W’ and ‘S’ keys on the keyboard to move forwards and backwards through the scene on the X and Z planes. The ‘A’ and ‘D’ keys are used to rotate the view to the left and right respectively. The ‘I’ and ‘K’ keys are used to look up and down. The user may move freely through objects and will not be able to move in the vertical direction.

# Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Input | Expected Output | Actual Output | Pass / Fail |
| 1 | Open program. | Scene loads with a grid of cubes for ground, some tetrahedron ‘rocks’ and some ‘trees’ made from cylinders and spheres in the correct orientation. Only the cubes will be in the same spot ever time since the other scenery has a random element to it. | [See Figure 1.](#_Figures) | Pass |
| 2 | Press the ‘W’ key on the keyboard. | View moves forward through the scene | [See Figure 2.](#_Figures) | Pass |
| 3 | Press the ‘S’ key on the keyboard. | View moves backwards through the scene. | [See Figure3.](#_Figures) | Pass |
| 4 | Press the ‘D’ key on the keyboard. | View rotates to the right. | [See Figure 4.](#_Lessons_Learned) | Pass. |
| 5 | Press the ‘A’ key on the keyboard. | View rotates to the left. | [See Figure 5.](#_Lessons_Learned) | Pass |
| 6 | Press the ‘I’ key on the keyboard. | View rotates up. | [See Figure 6.](#_Lessons_Learned) | Pass |
| 7 | Press the ‘K’ key on the keyboard. | View rotates down. | [See Figure 7.](#_Lessons_Learned) | Pass |
| 8 | At least 6 different shapes are used. | Scene is made up of a 50x50 grid of cubes, 15 trees made of 3 spheres and a tube each,  23 unbalanced tetrahedron rocks and a cylinder log. | [See Figures 1-8](#_Lessons_Learned) | Pass. |
| 9 | At least 6 different transforms used. | Cubes are drawn via a series of 6 transforms, Cylinders are drawn using 3 transforms and 3 shapes. The scene is generated via individual shapes and transforms assigned in a loop, while the entire scene is translated and rotated to give the illusion of the camera view moving through the world. | [See Figures 1- 8](#_Lessons_Learned) | Pass |

# Figures

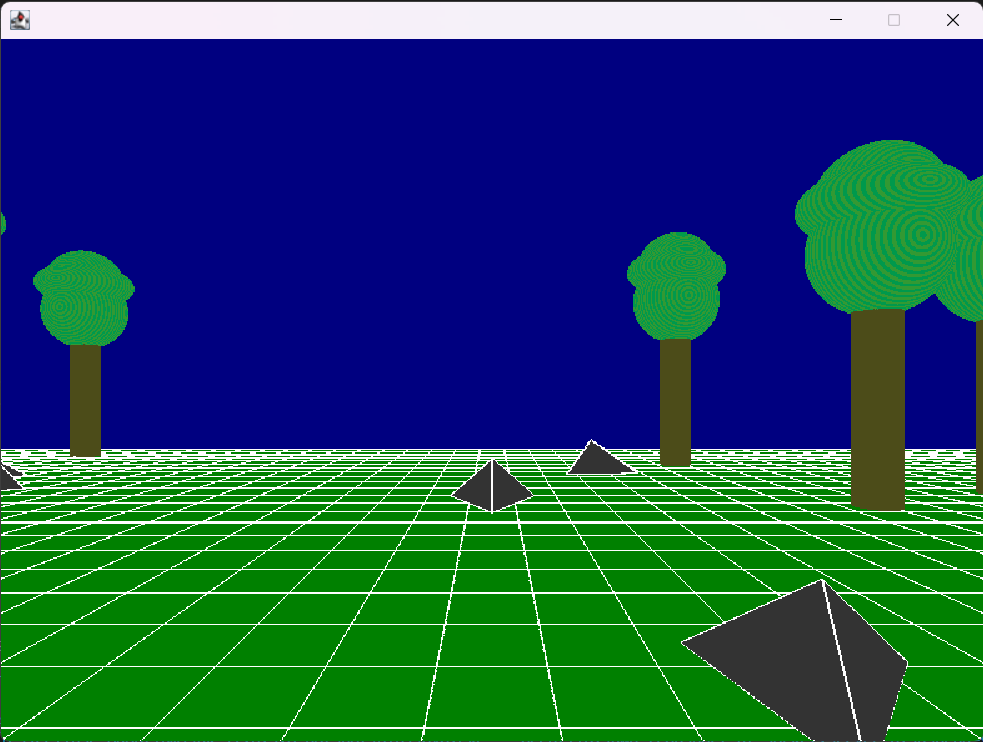


Figure – Origin

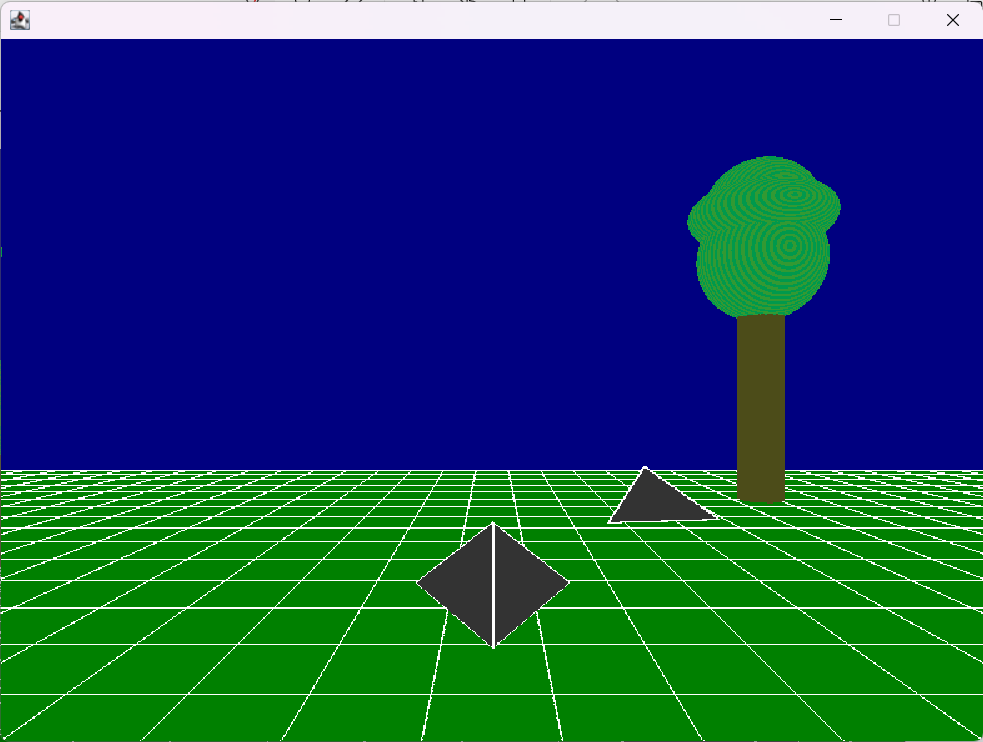


Figure Moving Forward

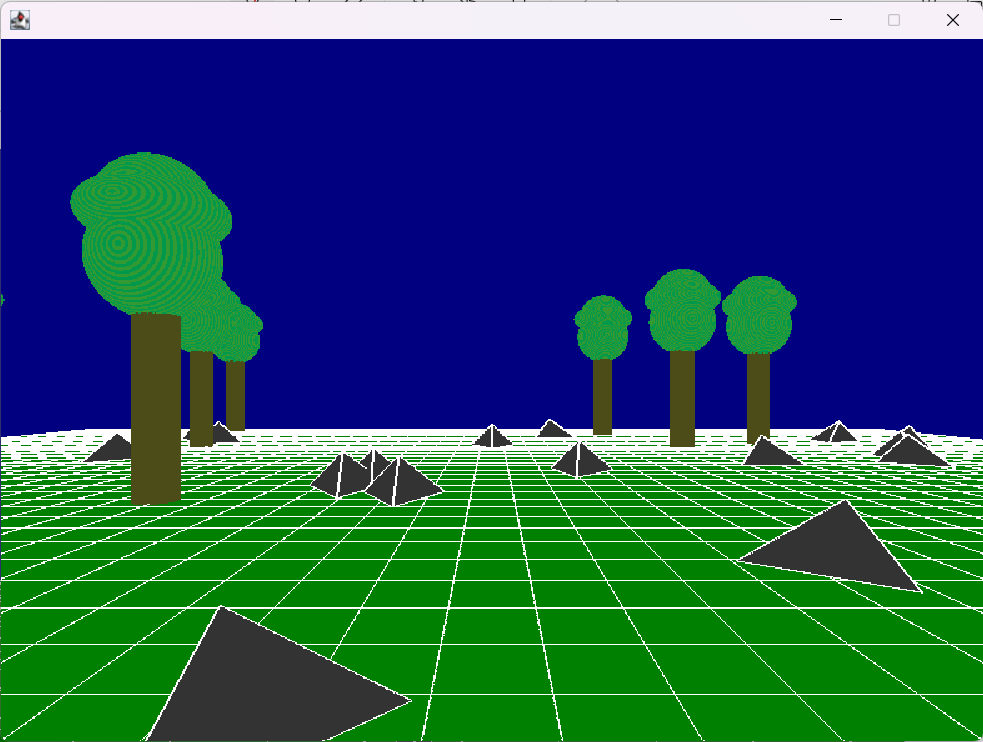


Figure – Moving Backwards

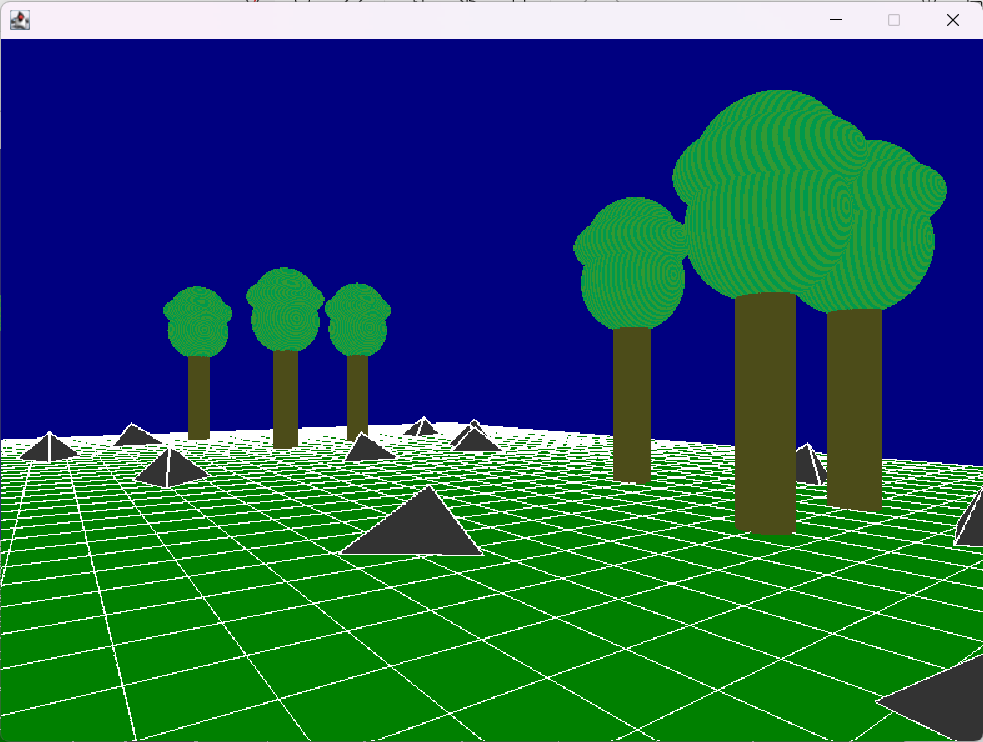


Figure - Looking Right

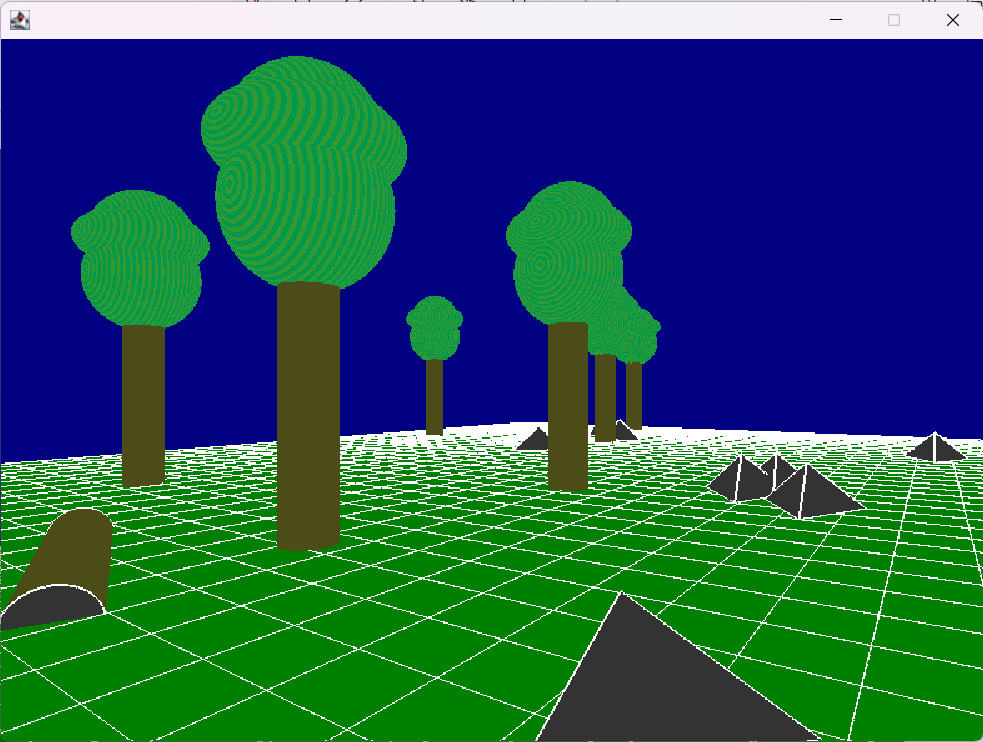


Figure Looking Left

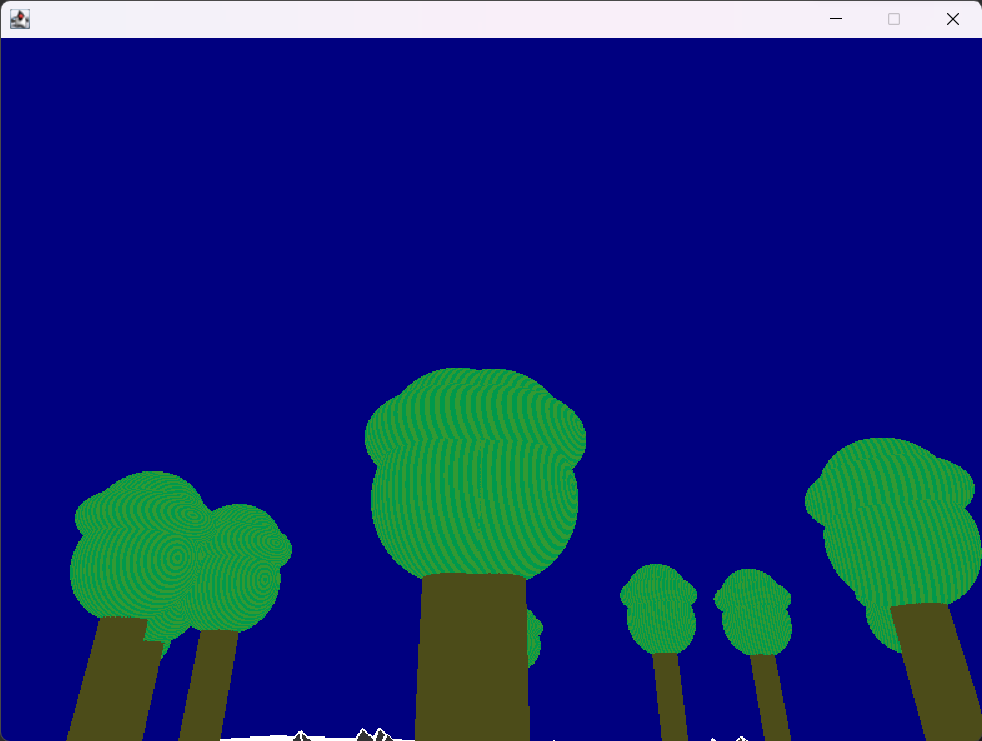


Figure - Looking Up

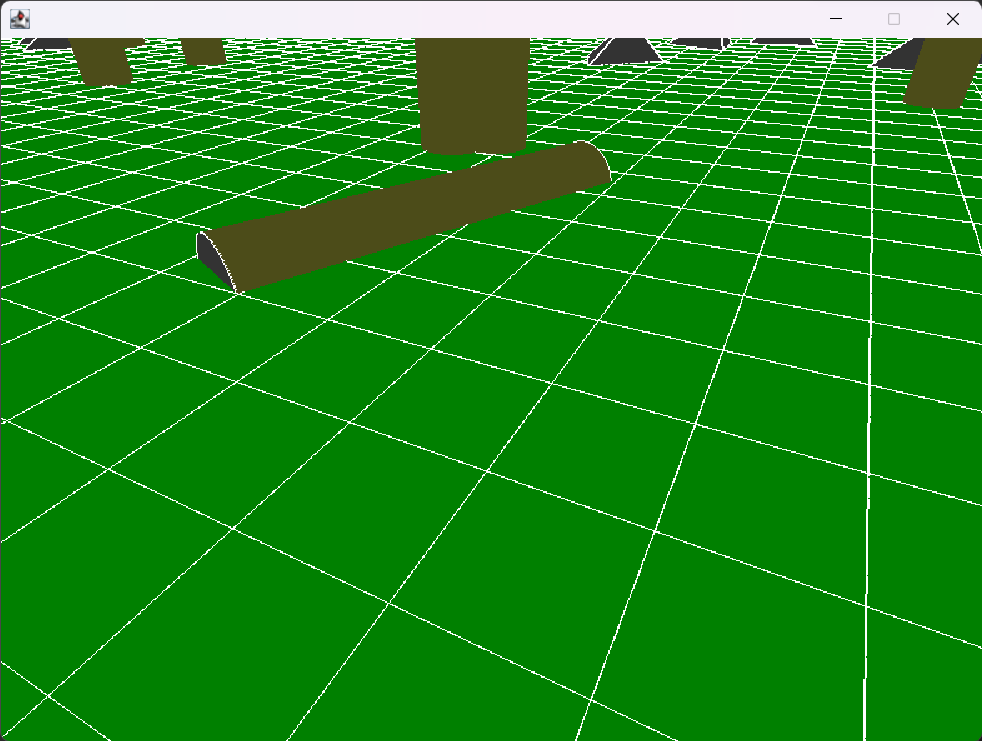


Figure - Looking Down

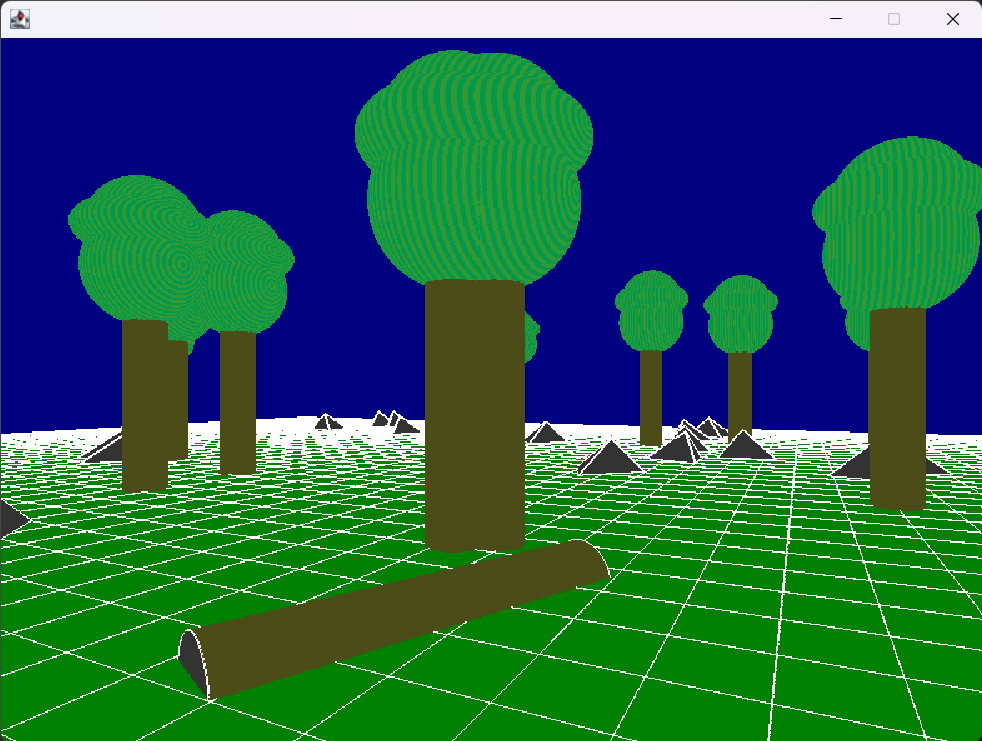


Figure - Log, Trees, and Rocks on the Ground

# Lessons Learned

1. I know a whole lot less than I thought about drawing round 3D objects. I had to rely on the code provided from the textbook in a few cases just to get the project up and running so I could start creating objects of my own.
2. Combining effort from another class into learning for this one took a lot of my time. I reused a lot of code from CMSC 335 in order to expand on the shapes I had created, and it seemed logical to include them in this project. Honestly, I would have been better off using Java2D for the other class/loading 3D images and then working on this one with it’s own set of classes.
3. I ended up modifying the geometry.Shape subclasses so many times that I really learned a lot about the draw order, pushing and popping from the stack, and how to achieve local transformations in a way I wanted.
4. I would like to have created smoother controls, a conditional drawing operation (to get better performance), bounds checking/collision detection, gravity simulation and a jump key. The way I went about drawing doesn’t really allow me to have a good way to control these things, since each object that was shown was instantiated and stored within an arraylist. I think a 3D tile map (just a 3D array of numbers) could work a lot better with each object only stored once (in a hashMap) and drawn many times based on the parameters set.