For this program, I used JavaFX and Eclipse IDE. Because JavaFX natively has their origin at the top left of the screen as opposed to the bottom left, the image will appear upside down. A 1000 X 1000 resolution was too big for my computer, so I used 800 X 800, which means Vsx, Vsy, Vcx, and Vcy were all made equal to 400. However, at the start, I kept s = 50 cm, and D = 2.5 cm.

For the beginning of my experimentation, I used the image/object and lines given. I transferred the 3D line coordinates to an input file of start-points and endpoints. Unless otherwise stated, the input file below is what I used for all experimentations.

A screenshot of a cell phone

Description automatically generated

Using the perspective projection algorithm and displaying the lines in an 800 X 800 resolution resulted in the output below.

A screenshot of a social media post

Description automatically generated

The image above looks to be very close to the screen and stretched out. Also, I wondered why it was so close to the bottom right corner. I figured it could be due to adding the viewport center (Vcx, Vcy), so I tested out removing Vcx and Vcy from my perspective projection equation.

A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

The results confirmed my thought, the image was to the top-left corner. I concluded this was because Vcx and Vcy represent the origin, so the image before was in the bottom right corner due to that section being the positive x, positive y quadrant.

I restored my program to its default values and experimented with translations. First, I tried to translate on the x axis. I wanted to translate the image to the left, so I translated by -100. Result below.

A screenshot of a cell phone

Description automatically generatedA picture containing screenshot

Description automatically generated

The resulting view was from directly in front of the image. Now there was a head-on view of the object. I knew that the pyramid on the bottom of the object would be out of view if this were a solid object, so I attempted to raise it by translating along the y axis. I applied a y translation of -200 to my previous translation (result shown below). The object appeared raised and the front side of the pyramid became visible.

A close up of a logo

Description automatically generated

Finally, maintaining the same x and y translations as before, I decided to apply a z translation by 100. This pushed the object back by 100 units on the z axis. However, in the result shown below, the smaller object appeared more clear and defined. I believe this is because the further away the point of view of the object is, the less stretched out it appears. (Whole window view on the left, clearer close-up view of object on right)

A screenshot of a cell phone

Description automatically generatedA close up of a logo

Description automatically generated

Through my experiments with translations, I observed that it appeared as if the object was stationary in the world coordinate system, and translations only served to move the virtual screen or viewpoint in front of the object. From this, you are able to get different perspectives and viewing angles of the stationary object.

For scaling, I decided to keep the following translations for my base object: x by -100. Y by -100, z by 25. This way I would have a neutral view of the object to experiment with. I started by scaling the x values of the object by 4. The base object and scaled object are shown below.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

I noticed from the results that scaling the x values appeared to change the width of the object. The scaled object appears like a stretched out version of the base object. I assumed that scaling y would change the object’s height or stretch it vertically, so I tried that next. Below is the result of scaling the y values of the base object by 4.

A screenshot of a cell phone

Description automatically generated

The base object’s height was definitely increased, stretched out vertically. I believed the last two images only appeared stretched out because I changed the scale of x or y, but kept the other the same. So next I tried scaling both x and y by 4.

A close up of a logo

Description automatically generated

The result confirmed my suspicions. In order to preserve the shape of the object, you have to scale x and y by the same number.

Finally, I tried scaling the object above by z. I scaled by 2, and assumed that doing so would make the object appear smaller.

A screenshot of a social media post

Description automatically generated

I was correct. By scaling z by 2, the object was now appeared 2 times smaller than it did in the image before. I concluded that, because scaling means multiplying the coordinates by your chosen scaling factors, you could easily drastically change the original object’s shape by scaling and increasing its height, width, or size (distance).

At this point, I decided to play with scales and translations to see if I could get a clearer 3D image. After translating x by -100, y by -100, z by 400, then scaling x and y by 16, I ended up with the object below. The 3D effect of the object was much more pronounced, due to first pushing the object far away (creating a better viewing distance), and then scaling the x and y values by a large number to increase the object’s size.

A screenshot of a cell phone

Description automatically generated

At this point, I attempted a rotation. I rotated the object above about the z-axis by 90 degrees and got the following result:

A screenshot of a cell phone

Description automatically generated

When rotating by 60 degrees about the z axis, I got the following result:

A picture containing screenshot, game

Description automatically generated

However, when rotating about the x or y axis, I got weird results. The following is me rotating about the x axis by 270 degrees:

A picture containing game

Description automatically generated

My y axis rotation looked similar. I believe I implemented them in the same way I did my z axis rotation, which leads me to believe I implemented it right, but rotations just look weird in 3d space. I tested rotations by 180 degrees and received the starting image back.

To test if my program would work on a different 3D object, I removed the bottom pyramid from default object. My input.txt now looked like this:

A screenshot of a cell phone

Description automatically generated

The result of that (without any transformations) on next page. The outcome was a regular 3d rectangle.

A screenshot of a social media post

Description automatically generated

From here, I decided to use the same translations (x by -100, y by -100, z by 400) and scales (x by 16, y by 16) to get the following.

A screenshot of a social media post

Description automatically generated

From here, I decided to try changing the parameters of D and s in the perspective projection equation. I started with D, changing it from 2.5 to 1:

A screenshot of a social media post

Description automatically generated

The result was a visually smaller object. I thought this was confusing because D is supposed to be distance from screen, so I figured that lessening the viewing distance would lead to a larger image. However, it makes sense that it would be smaller, because D is in the numerator of the perspective projection equation. Therefore, smaller numbers would lead to smaller objects.

Next, I kept D back at 1, and changed s to 25, I figured the object would be bigger this time:

A screenshot of a social media post

Description automatically generated

I was correct, the object was bigger due to s being in the denominator. Therefore, a smaller number would lead to a bigger object.