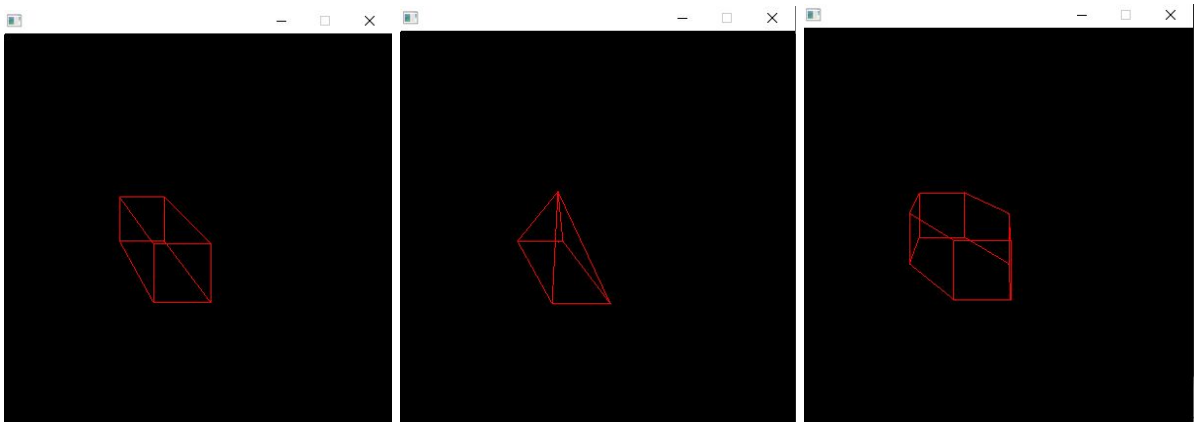
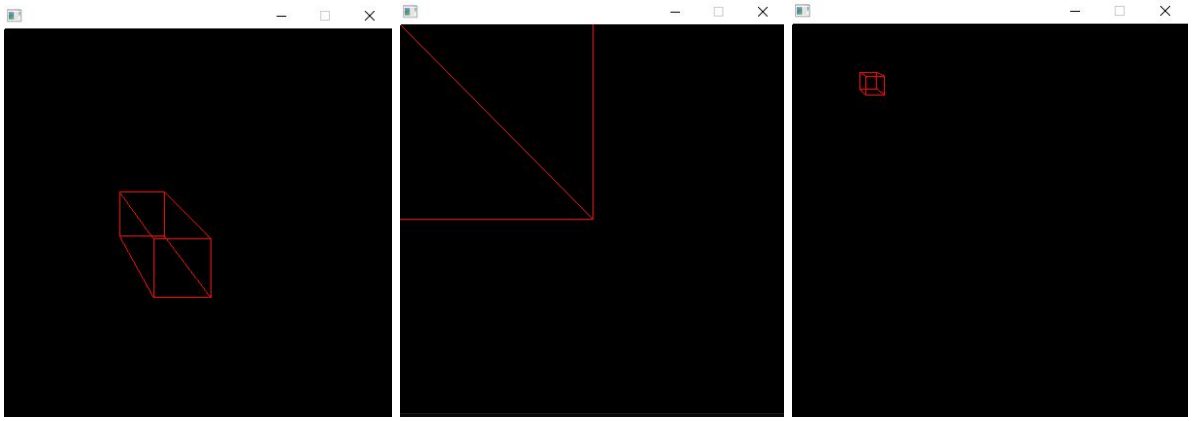
Digital Imaging Assignment 4 Writeup
Danielle Mawson

In our latest project, we were responsible for implementing transformations for 3D objects and projecting them on a 2D viewport. I have defined a cube, a pyramid, and a hexagonal prism stored in in1.txt, in2.txt, and in3.txt respectively. I created each line by defining 2 points, each with 3 coordinates (x, y, and z) and read it into our Assignment 1 line drawer. When I first viewed by cube, I thought I had a problem. Because many of my points line up along the coordinate axes, I expected to be viewing my object as if in orthogonal view. Each of my object's centers are positioned at (0,0,0). However, I saw it from an angle before I even completed any transformations. Initially I thought something was wrong with my code, but then I remembered the arbitrary position of the Eye Coordinate System was (6, 8, 7.5). It was not the object that was at an angle, but rather my eye!



in1.txt in2.txt in3.txt

Out of curiosity, my foolish first experiment was to move the viewport to position (0, 0, 0). Because of this, I saw a huge, single angle of the cube in the fourth quadrant. I'm too close now! When I set the z coordinate of the eye coordinate system back to 7.5 and the same issue persisted. Because the cube's negative x and or positive y points were clipped off, we still could only see the points in the cube in the fourth quadrant. However, I was able to see points in a more orthographic view. I can see two connected squares due to the effects of perspective projection. The part of the cube in the back appears smaller than that in the front. Changing my viewpoint coordinates effects what is being clipped from the image. I was able to verify to myself that my objects are okay and am ready to begin my experiments in the order listed (different images, changing parameters (position of VP, screen size, distance from screen, number of pixels, different transformations, etc.).



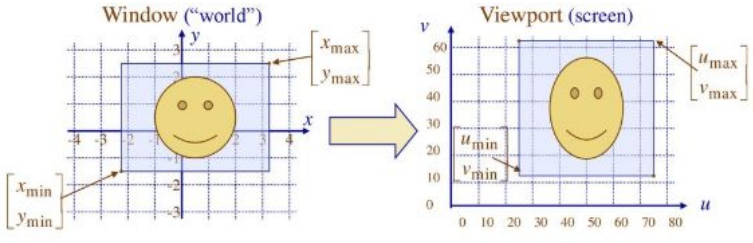
View point at (6, 8, 7.5) view point at (0, 0, 0) viewpoint at (8, 6, 20)

Changing my screen size (number of pixels to the center) will affect the window to viewport transformation.

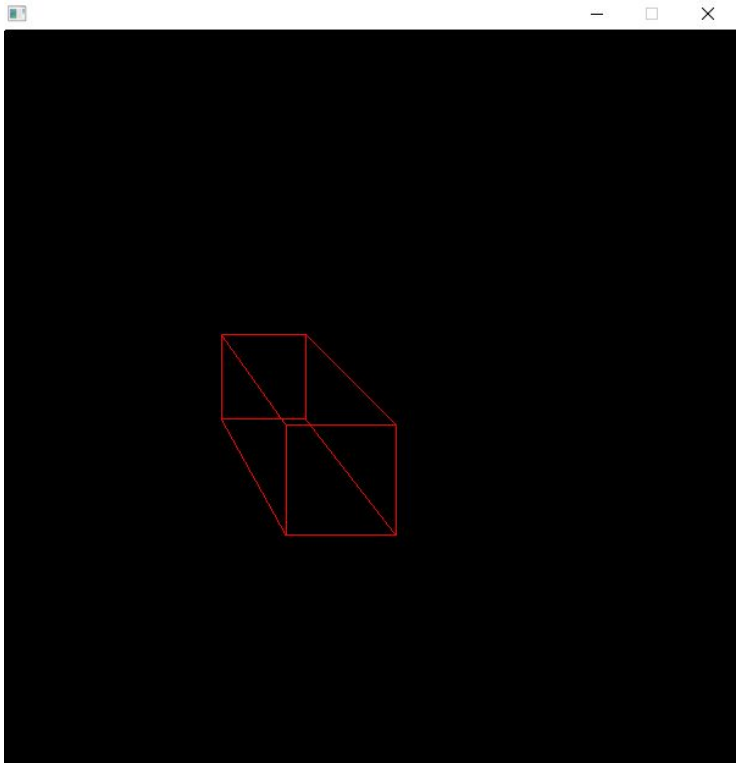
$$t_x = -x_{\min} \text{ and } t_y = -y_{\min}$$

$$s_x = \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} \text{ and } s_y = \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}$$

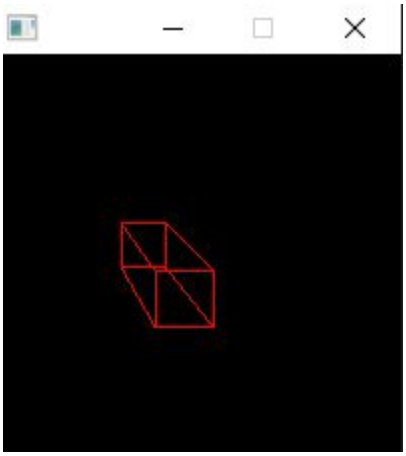
$$t_x = u_{\min} \text{ and } t_y = v_{\min}$$



If we increase the delta u value, we increase the value of each point's x coordinate. If we increase the delta v value, we increase the value of each point's y coordinate. When I change the screen size, it reminds me of a scale transformation. If I were to make the screen shorter in the x direction, the lines seem to bunch together. If I were to make it longer in the y direction, the lines seem to stretch upwards. Because our dimensions are in the shape of a square, I used one variable for all four sides. Because of this, everything shrinks and grows uniformly.

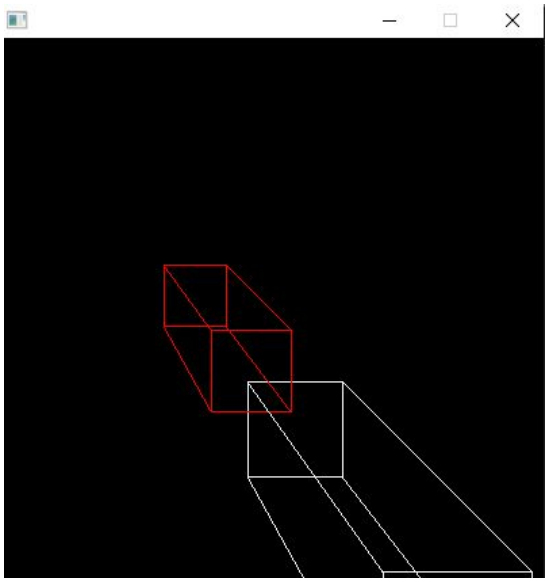


screen size 300 (not to scale; slightly smaller)

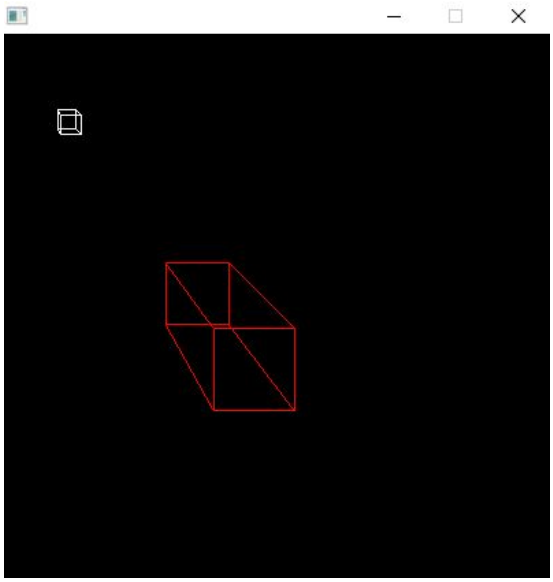


screen size 100 (to scale)

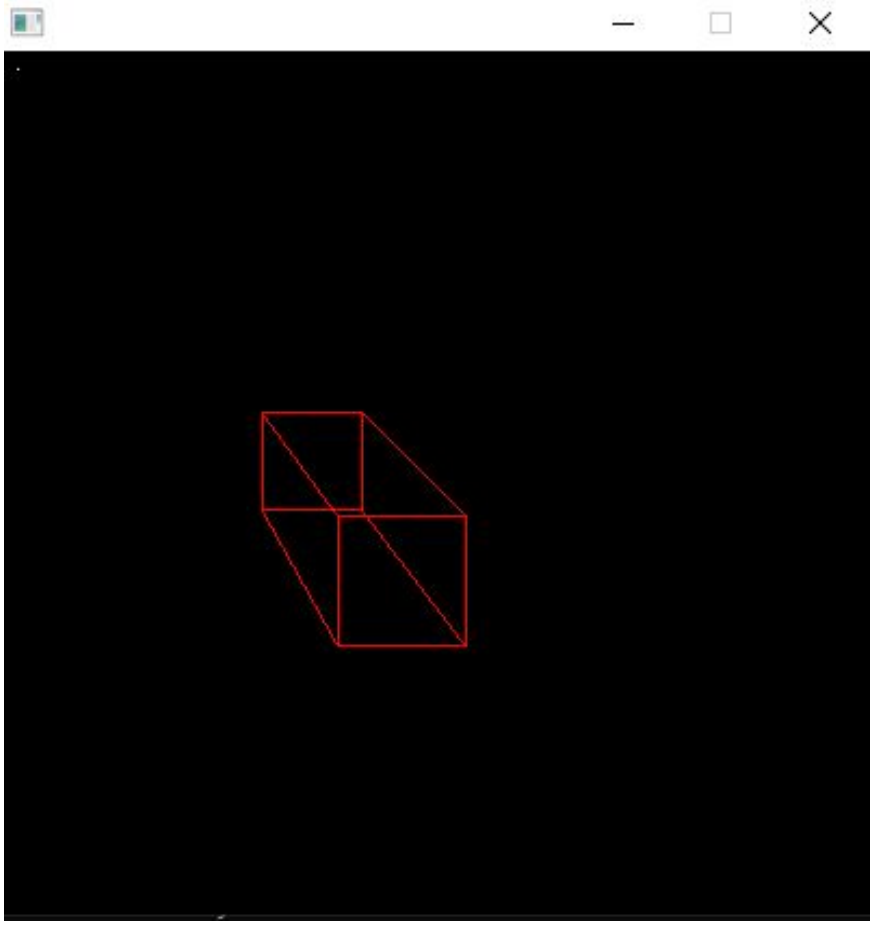
The distance of the screen changes the object rather intuitively. If you bring the object closer to the screen, in the positive z direction (but less than 7.5) the object appears larger. This makes sense with our daily experience with perspective. As you push the object farther away from the screen, in the negative z direction, the object becomes smaller and smaller. When pushed to an extreme, it appears as a dot in the perspective's vanishing point.



object translated by 2 units (z direction)



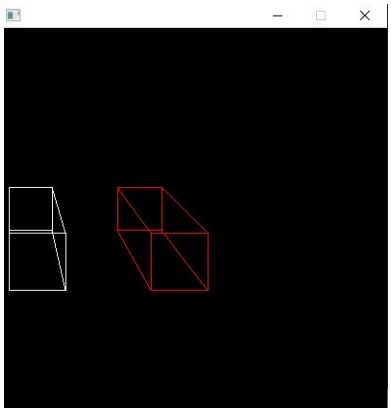
object translated by -20 units (z direction)



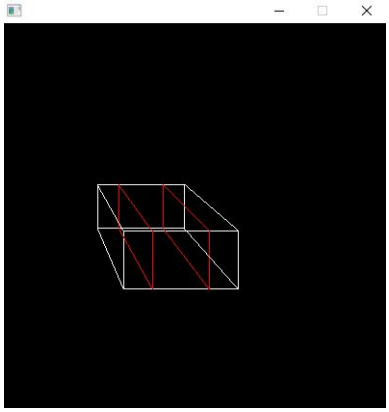
object translated by -9999 units (z direction)
Enlarged so you can see the dot (upper-left corner)

Because it is easier to visualize, I used by cube object for the above examples, but to demonstrate my transformations, I will use all 3 options available.

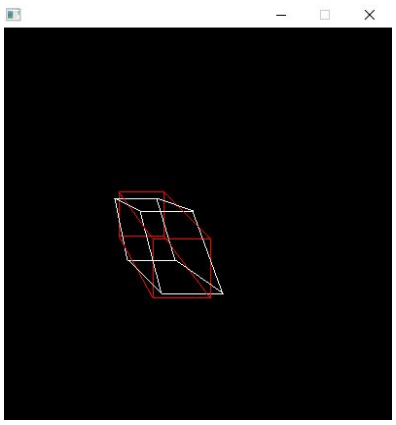
in1.txt:



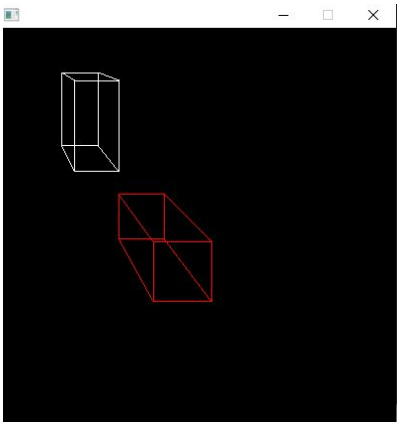
horizontal transformation of 5



horizontal scale factor of 2

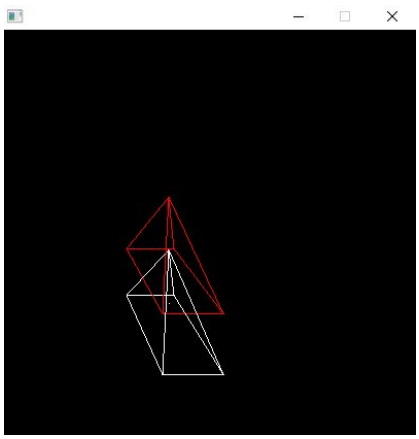


rotation by 30 about x-axis

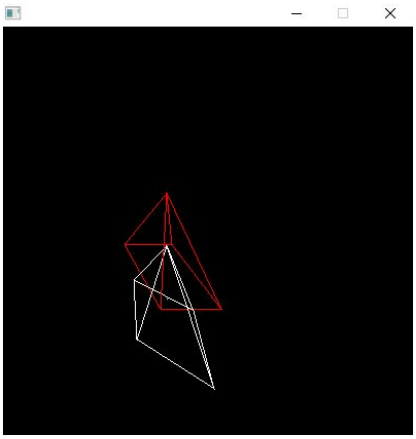


vertical scale factor of 2 about point (-2, -2, 2)

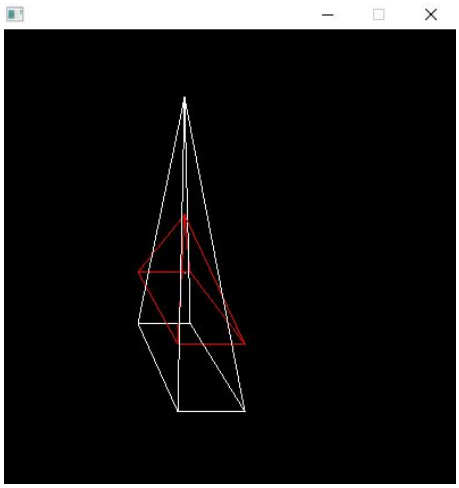
in2.txt:



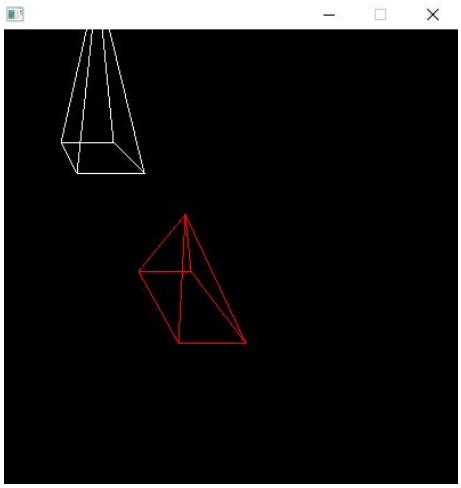
vertical translation of -2



rotation by 60 about y-axis
(concatenated with v.t. of -2)

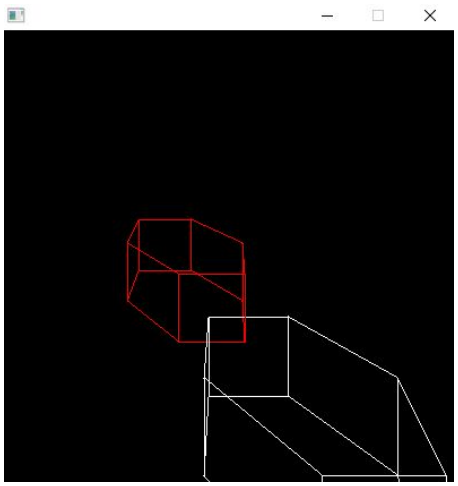


vertically scaled by 3 (basic (0, 0, 0)

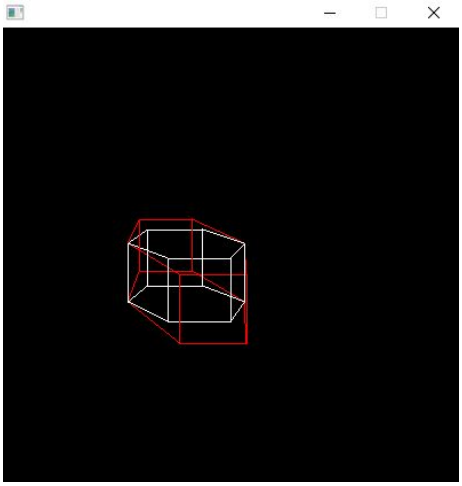


vertically scaled by 2 about (-3, -3, 0)

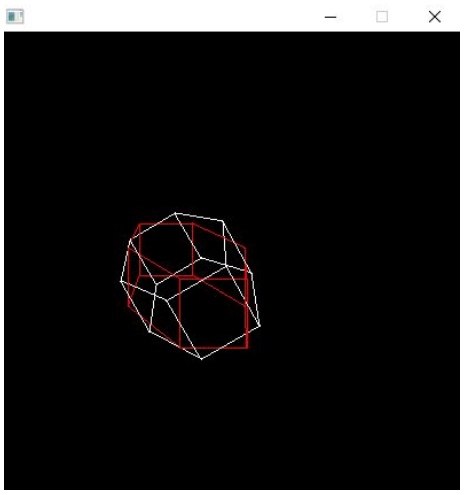
in3.txt:



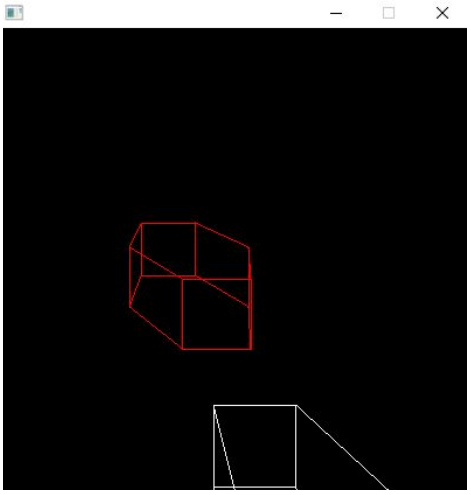
translated in the z direction by 3



scaled by 0.5 in the z direction



rotated by 30 about z axis



scaled by 2 in the z direction from point (0, 2, -2)