CHASE MELTON CSCI 4810

Computer Graphics: Perspective Projection

Introduction

Perspective Projection is the process of transforming coordinates defined in the World of Eye coordinate system to a coordinate system that can be displayed on a 2D surface, in this case, a computer screen. The program described in this report accomplishes this task through the various transformations required.

Process

The process starts with a custom-designed text file containing input representing an image. An example:

0 1 0 -> 0 1

1 0 0 1 2

0 0 1 2 3

-1 0 0 3 4

0 0 -1 0 2

0 3

0 4

4 1

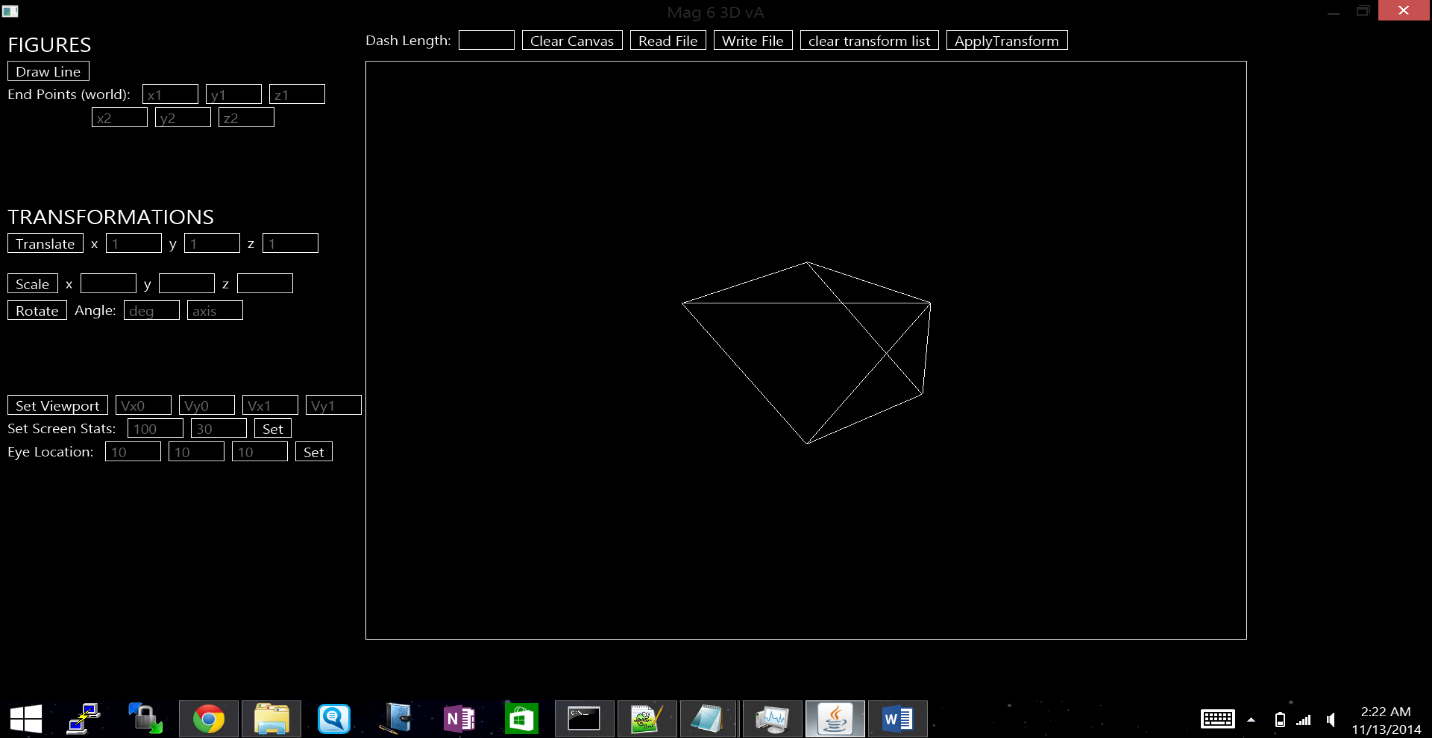
The first block of 3-digit sets of numbers represent the coordinates of an image (in this case, a self-designed pyramid) expressed in the world coordinate system. Following that block is a delimiter, letting the program know that it is finished reading coordinates. The second block of two-digit sets of numbers represent the coordinate pairs that are used to create lines. These are stored as reference in Line objects so that the vertices may be updated when necessary.

These coordinates are run through the Perspective Projection algorithm:

1. The supplied world coordinates are transformed to eye coordinates via the V transformation matrix. This process uses the user-supplied observation point, expressed in the eye coordinate system, to perform a series of transformations that convert the world coordinates to eye coordinates.
2. The eye coordinates are converted to clipping coordinates via the N matrix. This process involves user-supplied values for the ideal viewing distance (D) and the size of the screen divided by two (S).
3. Clipping coordinates are converted to screen coordinates using additional values for the viewport size. Unlike the observation point, the viewing distance, and the screen size, the viewport size has hard-coded default values representing the size of the drawing canvas (in this case, 1180 X 845). The user may change these values through the interface without modifying the code.
4. The resulting screen coordinates are drawn onto the screen using the JavaFX PixelWriter Interface and Bresenham’s algorithm to calculate the lines.

Experimentation: The Observation Point

Perspective Projection relies on an “observation point” to display an image to the computer screen. Further readings have even described this point as a “camera” into the virtual world being displayed. The reason for this is that modifying the observation point modifies the image in a predictable way. For instance, Figure 1 is the before mentioned pyramid displayed in the program:

Figure 1

This pyramid is displayed using

a viewing distance of 100, a screen size of 30 (S = 15), and an observation point of (10, 10, 10). This picture is effectively displayed about the “origin” (based on the world coordinates), so increasing any of the values of the observation point should result in an object displayed “further away.” However, these will change the perspective on the object as well, not just the size.

Figure 2 shows the pyramid displayed at (10, 10, 50).

Figure 2

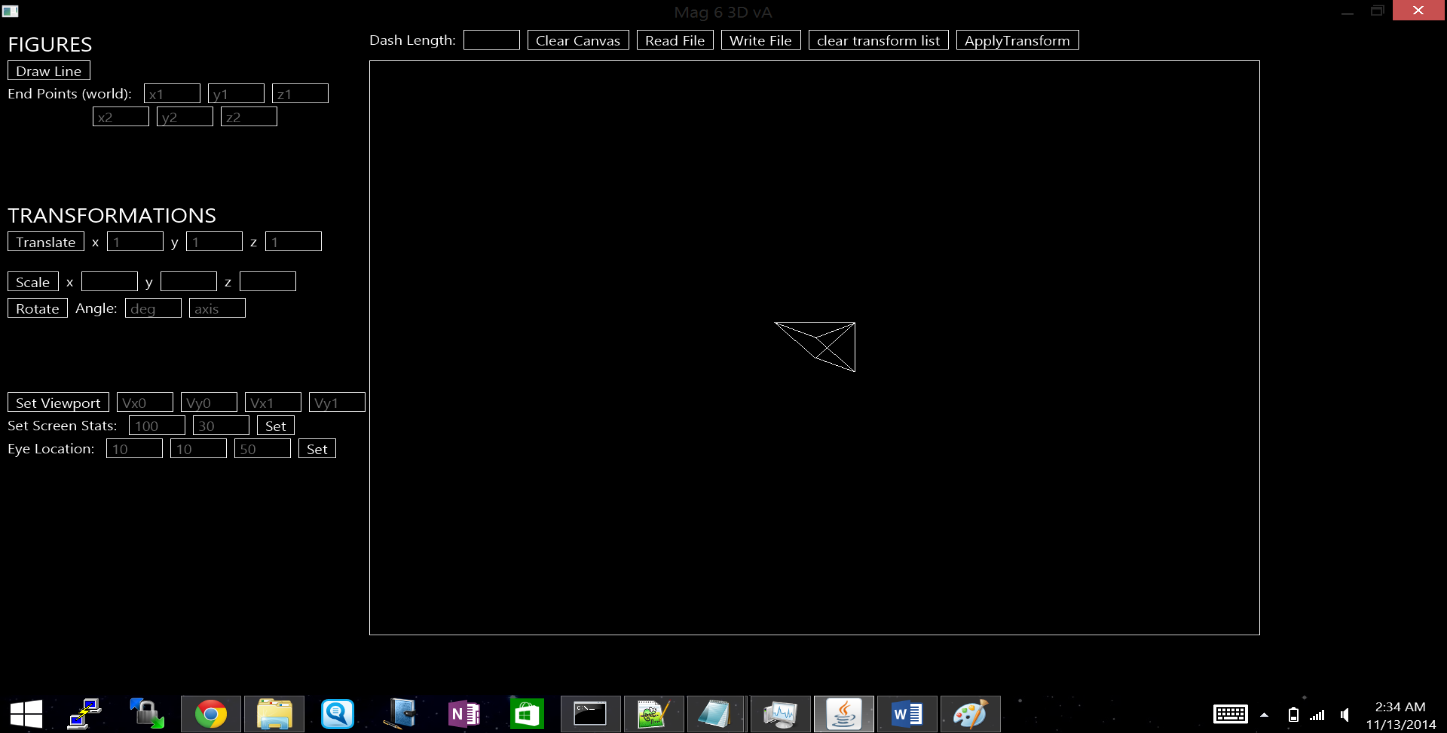
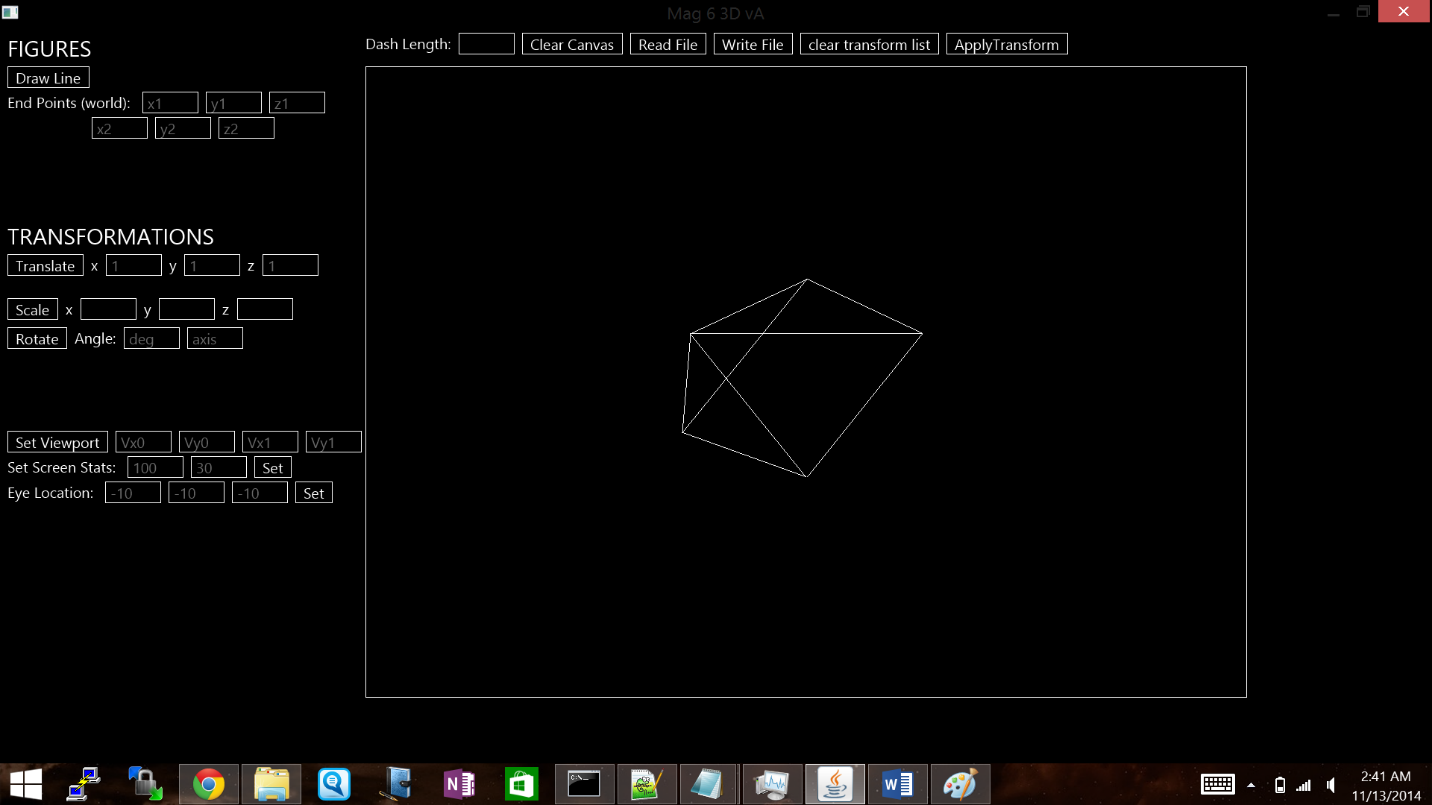
Moving “away” in the z direction does not just make the image smaller. It is being viewed from an angle already, since the X and Y values of the observation point are both 10, and therefore, the actual perspective changes as well as the size of the object. The viewing angle becomes less extreme, and thus there is a sense of having “moved left” in terms of the perspective we now have of the pyramid.

Figure 3 uses (-10, -10, -10)

Figure 3

as the observation point. When compared to Figure 1, the pyramid now appears “flipped.” This is because the observation point is now set in the opposite direction from the original, giving a mirrored perspective the original pyramid. To summarize, changing the observation point does nothing to the actual coordinates of the image- it simply alters the perspective, making it appear further away or reflected from a similar image. This is what perspective projection is designed to do- fool the user into seeing the same three dimensional object and believing it has changed based on supplied information about where that object is being viewed from.

Experimentation: Ideal Viewing Distance and Screen Size

Figure 4

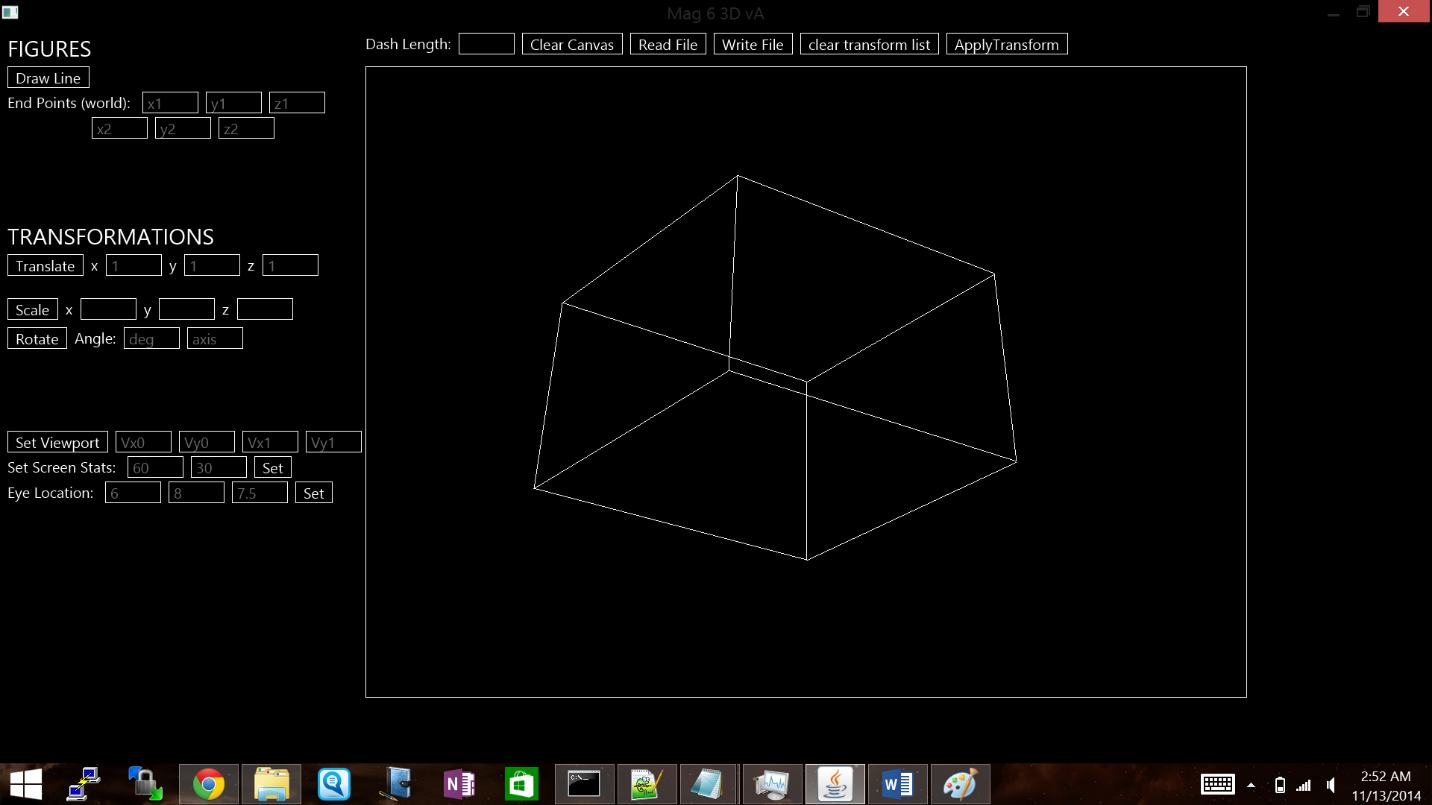
Figure 4 is the cube defined in the project description displayed on the computer screen. It is viewed using the values defined in the description as well: The Ideal Viewing Distance is 60 cm, and the screen size is set to 30 cm (the actual number used in calculations is the screen size divided by two, so 15 in this case). Let’s start with the viewing distance: it seems logical that increasing or decreasing the viewing distance will increase or decrease the size of the projection, respectively, in order to maintain the “same” projection. To test this hypothesis, Figure 5 shows the cube displayed with a viewing distance reduced to 10 cm.

Figure 5

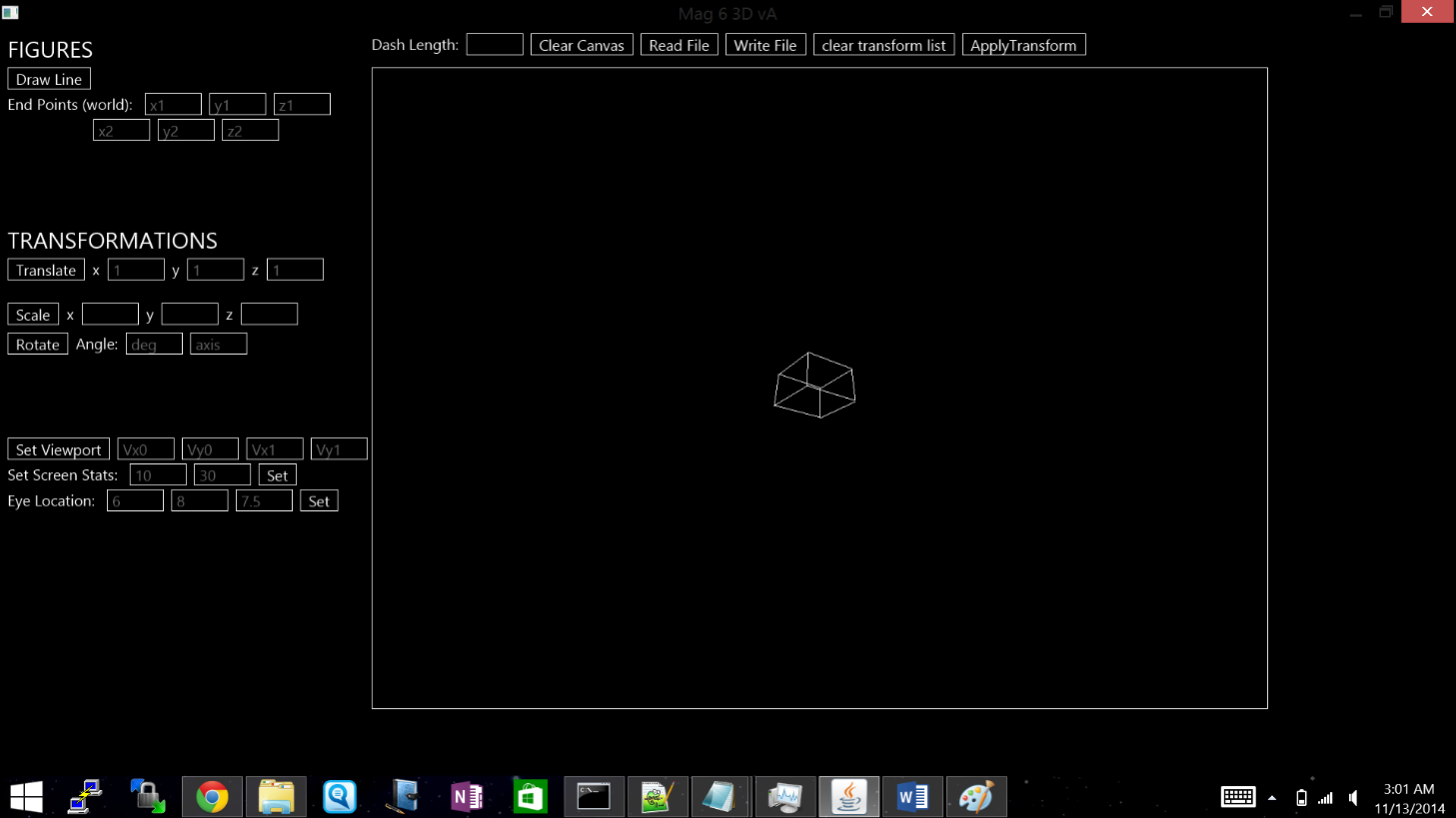
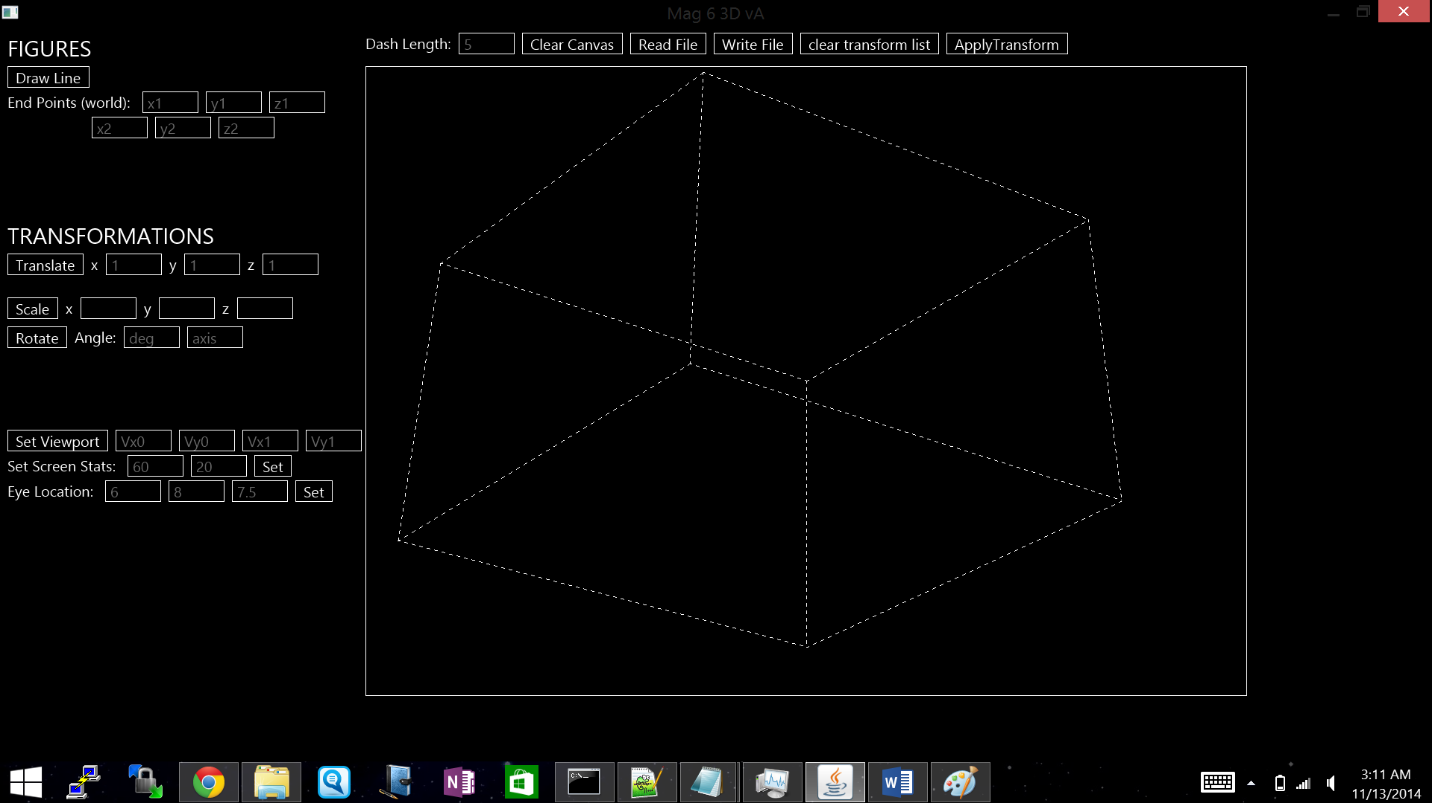
Indeed, the cube has “shrunk.” A shorter ideal viewing distance means you *should* be closer to the screen when you look at the object. When adjusting viewing distance, one would see that the projected object does not change in size. However, maintaining a fixed viewing distance will give the impression that the cube is now smaller (or larger, in the case that the viewing distance increases instead of decreases). Unlike modifying the observation point, modifying viewing distance (and screen size, as will be shown) does not change the viewing angle, meaning the resulting projection is still seen “straight on” in terms of the original image. Again, if one does not adjust their distance from the screen, only a zooming effect will be noticeable. The visible parts of the image will not change.

Figure 6

Figure 6 resets the viewing distance to 60 and drops the screen size to 20. In comparison to dropping the viewing distance, dropping the screen size has the opposite effect. The cube now appears larger than the original. To view the same size object on a smaller screen, more area is needed to maintain the same perspective. Likewise, on a bigger screen, less area would be needed to maintain the same perspective. However, if the physical screen size remains constant while the size supplied to the perspective projection algorithm changes, the cube will appear to grow if the screen size drops, and will appear to shrink if the screen size increases.

Summary

Perspective Projection is designed to make the user believe they are viewing a three dimensional image on a two dimensional screen. It makes sense that changing various values will have different effects on the actual image displayed on the screen. If you walk around a statue in a park, you’re going to have a different perspective- see a different “image”- depending on where you’re standing at a given moment. Perspective Projection simulates those differences. If you view the front of an object from an observation point of (0, 0, 10), it makes sense that you’ll view the back (or at least a different area) from (0, 0, -10). If you change the viewing distance by walking away from the statue, it makes sense that the statue will appear smaller, although the actual size has not changed. Perspective Projection simply mimics these differences on the computer screen through algorithms that compute these differences.