Andrew Seba  
Math/Phys191

Scaling and Translations Lab

# Introduction:

For this lab we are scaling a group of vertices of an object. These methods go to the root of moving and manipulating objects for every game using vertices ever. This application can be used for a lot of things related to 3D relationships, even including modeling.

# Methods:

Matrix multiplication is the main part of this lab. The first method was straight translating the object. To do this we just need to add the change of each axis to the objects vertices.

The second method was doing a raw scale, by using a 4 by 4 matrix we are able to scale the object’s vertices. In the solution though I used just 4 Vector3D objects for the matrix to utilize the Vector3D dot product like this equation.

The third method uses a concatenated matrix to move the object back to center and then scale it then move back to its original origin. This matrix looks like this equation followed:

For the lab for each row in this matrix got declared as a new vector to utilize the dot product method of the vectors.

# Results:

Using a 3 vertex object with values:

<0, 1, 2>, <3, 4, 5>, <6, 7, 8>

## Translating:

Translating with 5 in the positive x 2 in the negative y and 0 in the z axis. Results in:

<5, -1, 2>, <8, 2, 5>, <11, 5, 8>

## Raw Scaling:

Raw scaling 2 in the x axis, 0.5 in the y, and -1 in the z results in:

<0, 0.5, -2>, <6, 2, -5>, <12, 3.5, -8>

## Center Scaling:

Center scaling 2 in the x axis, 0.5 in the y, and -1 in the z results in:

<-3, 2.5, 8>, <3, 4, 5>, <9, 5.5, 2>

# Conclusion:

The results show that the translating is pretty straight forward, the raw scaling moves the center of the object while its being scaled, and the center scaling shows that the scaling from the center will push out the object in all directions to keep the origin the same.

# Post-Lab:

1. For centimeters we need 3 decimal places, and for a float we get about 7 accurate decimal digits. By the 10,000’Th translation we would have lost a whole centimeter of information on vertices, because if we lost a .000001 on our vertex 10,000 translations later we would have messed up our centimeter. At 30 frames a second it would take 333 seconds for a visible error to occur, after 5 minutes we get our first visible error and if you’re playing on the same scene for more than that we might see lots of visual artifacts building up.
2. That way we calculate a rough center that might gain artifacts but the shape remains the same because it’s based off the center that’s taking the calculation hit.