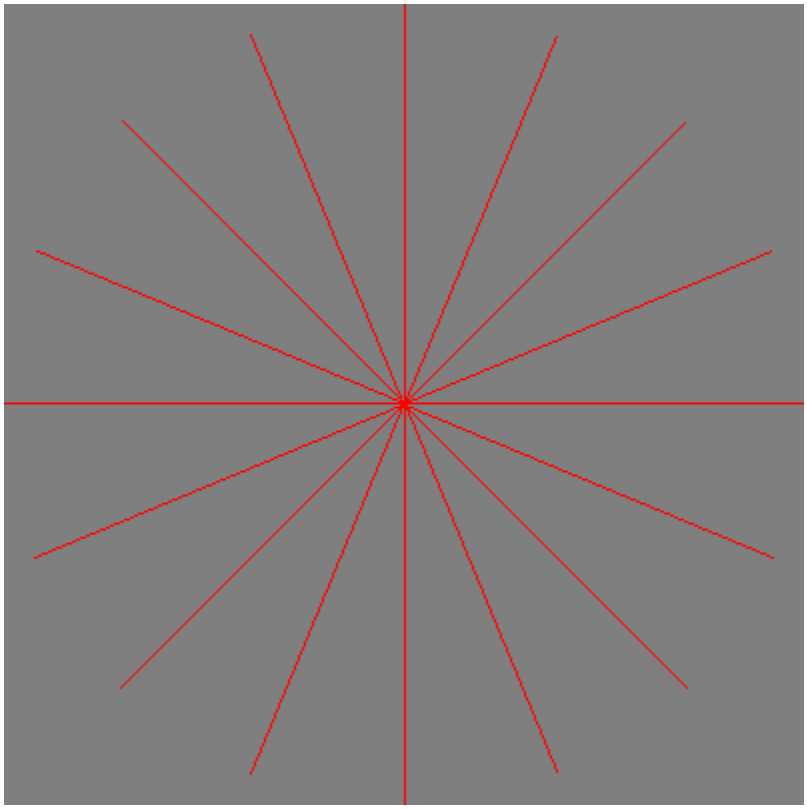
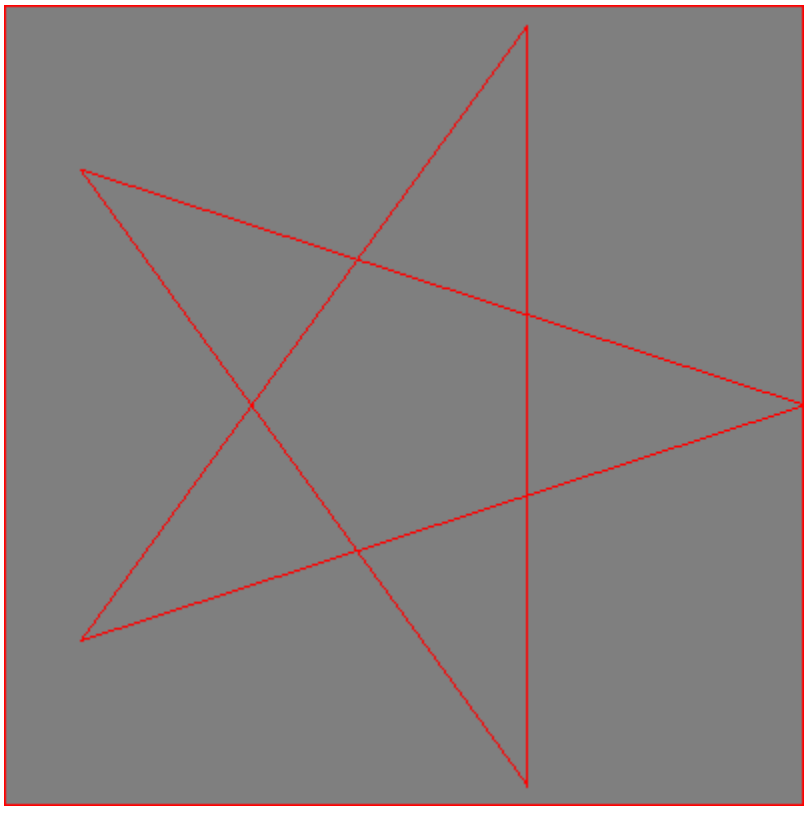
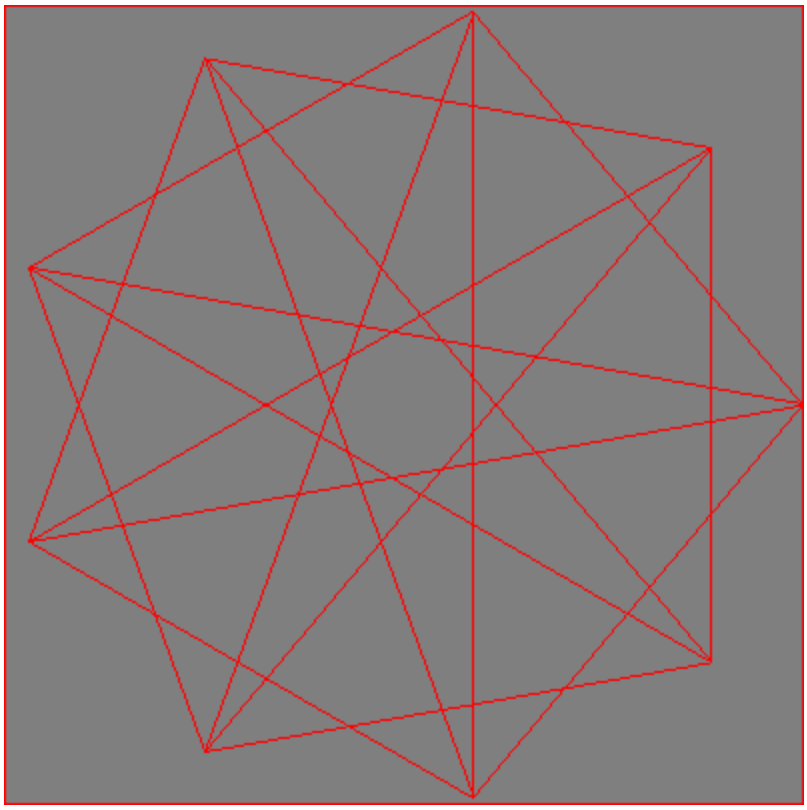
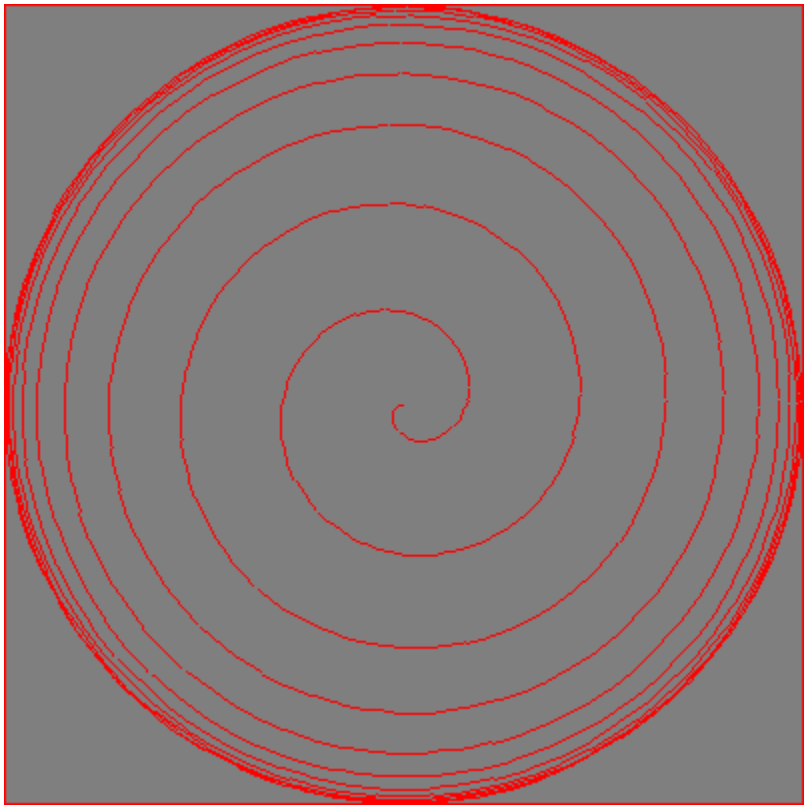
1 2D Line Rasterization

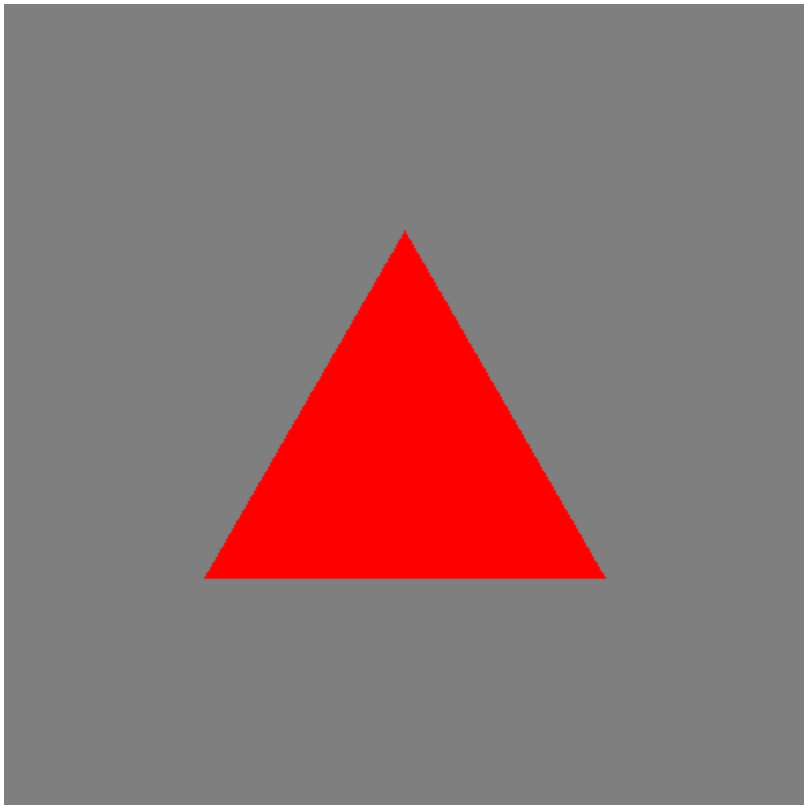


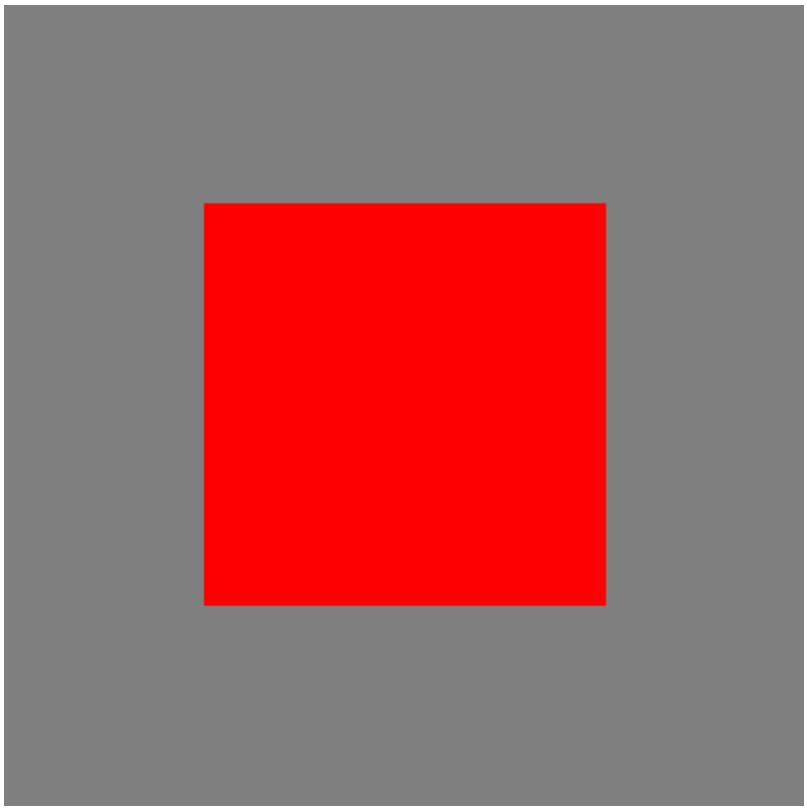


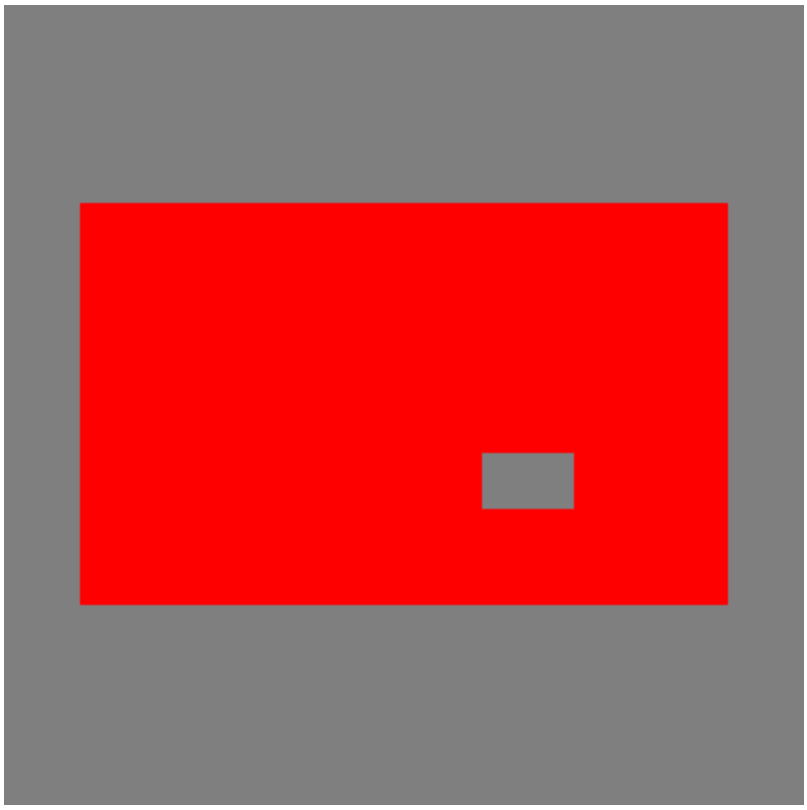


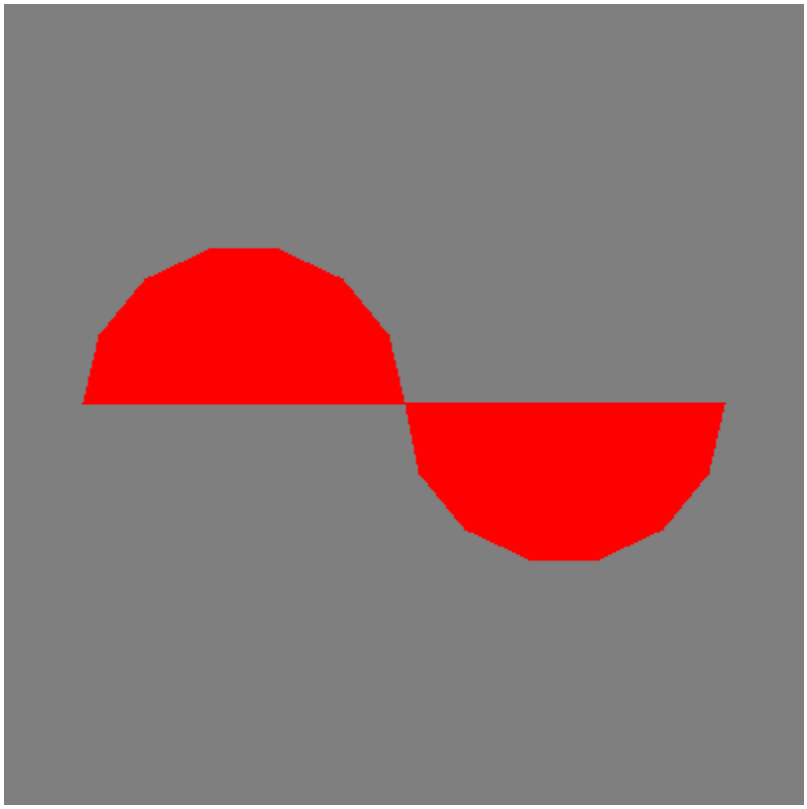


2 2D Triangle Rasterization

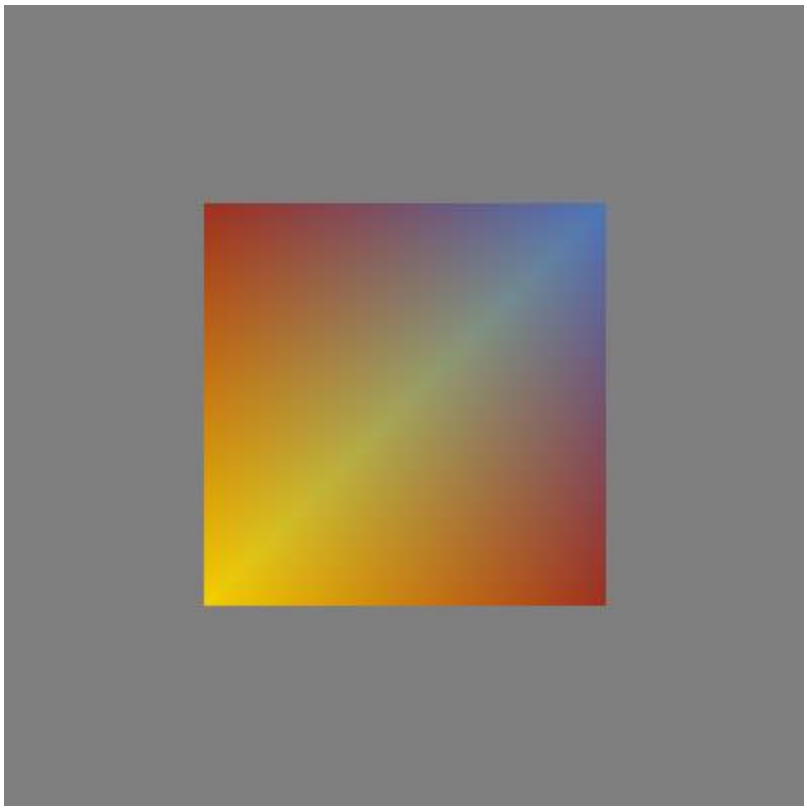


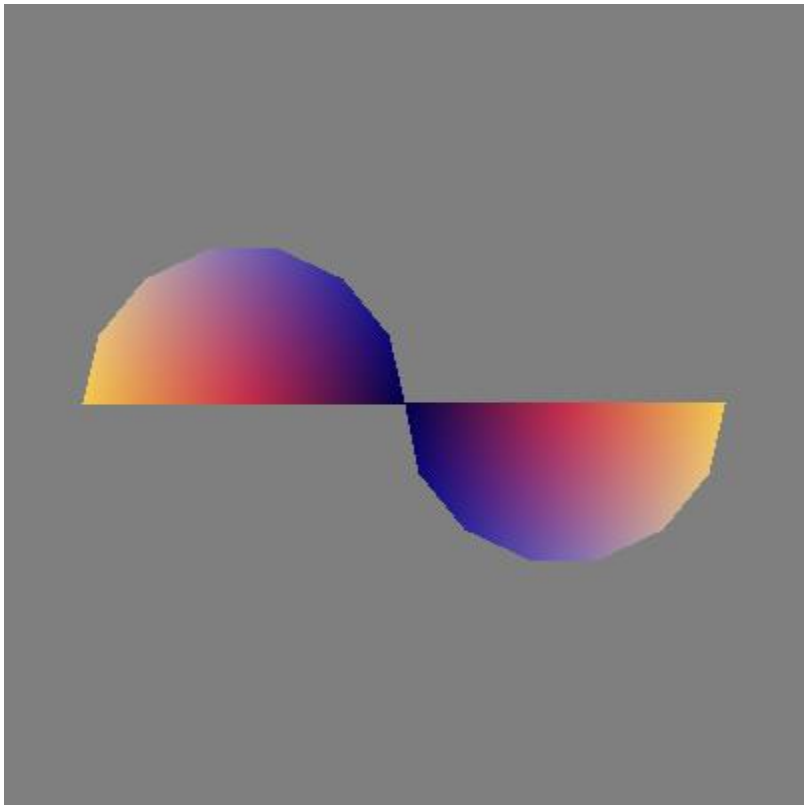






3 Barycentric Color Interpolation





Question 3:

The only difference between the two rainbow_triangle files is that the leftmost image has t 0 1 2 and t 2 3 0 while the rightmost image has t 0 1 3 and t 2 3 1. These changes shift where the first point of the triangle is located.

4 2D Transforms

Methodology:

The existing .pline and .tri files were as basis. Lines were added to these files to control the scale, translation and rotation transformations. The Line2F and Triangle2f structs in util.h were modified to accommodate these addition inputs. The methods load_polylines and load_triangles in io.cpp were also modified to bring these new values in from file. The Scale, Translate and Rotate methods were then added to draw.cpp.

To support transformations a .pline line is changed from:

polyline
-1.000000 0.000000
1.000000 0.000000
to
polyline
-1.000000 0.000000
1.000000 0.000000
0.500000 0.500000 – scale
0.250000 0.250000 – translation
1.570796 321.0000 – rotation and transform ordering right to left

This will take across the middle of the screen and scale it to ½ its original size. It will them move the scaled line from -0.5 0.0 0.5 0.0 to -0.25 0.25 0.75 0.25. Then then line will be rotated pi radians counterclockwise around the origin for a final line of -0.25 -0.25 -0.25 0.75. the 321 defines the order of the transforms from right to left. In this case 1 – scale, 2 – translate, 3 – rotate.

To support transformations a .tri triangle is changed from:

p 0.000000 0.433000
p -0.500000 -0.433000
p 0.500000 -0.433000c 0 .63 .19 .13
c 1 .63 .19 .13
c 2 .63 .19 .13
t 0 1 2
to
p 0.000000 0.433000
p -0.500000 -0.433000
p 0.500000 -0.433000
c 0 .63 .19 .13
c 1 .63 .19 .13
c 2 .63 .19 .13
s 0.500000 0.500000 - scale

m 0.250000 0.250000 - translate
r 1.570796 321.0000 – rotate and transform order from right to left
t 0 1 2

This transform of the triangle is analogous to the transform of the line described above.

Question 4:

Scale triangle to .5

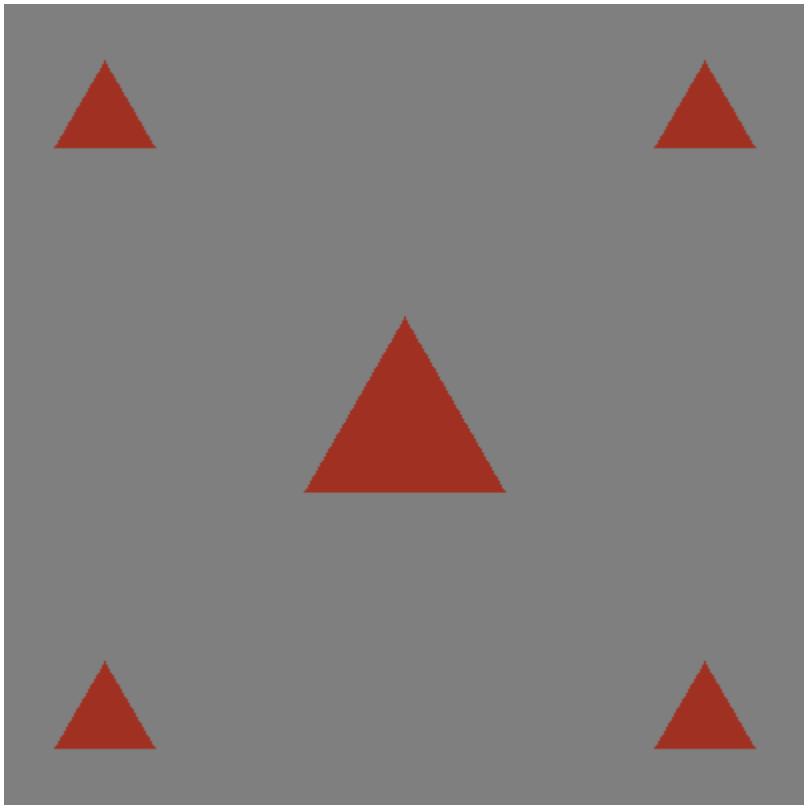
Draw in red at center

Repeat 4 times

Scale triangle to .25

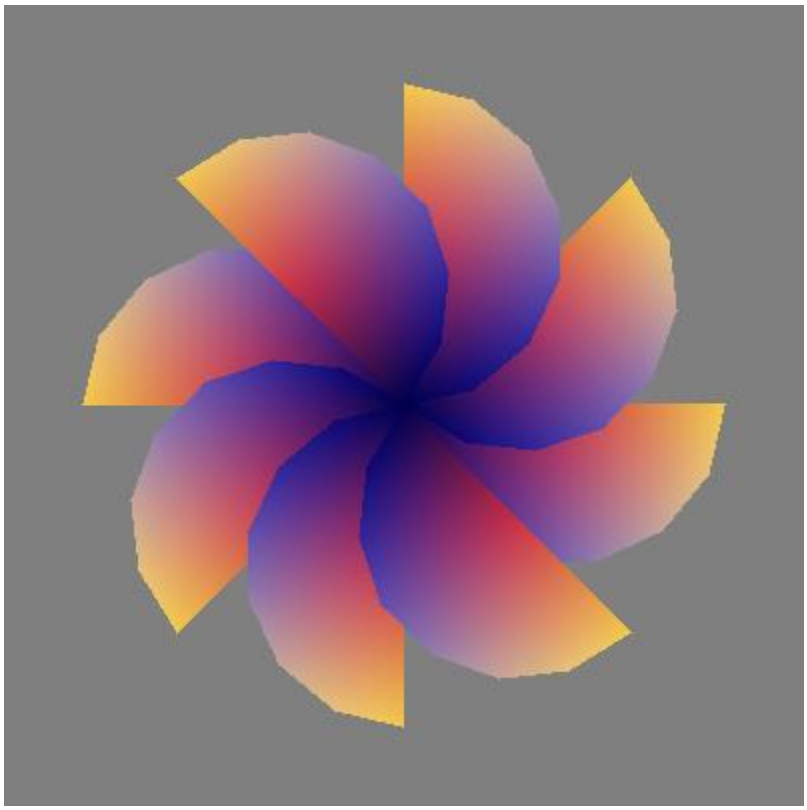
Move to x = .75 y = .75 set signs based on corner

Draw in red.



Question 5:

Draw triangle_fan 4 times rotated 0, pi/4, pi/2 and 3pi/2.



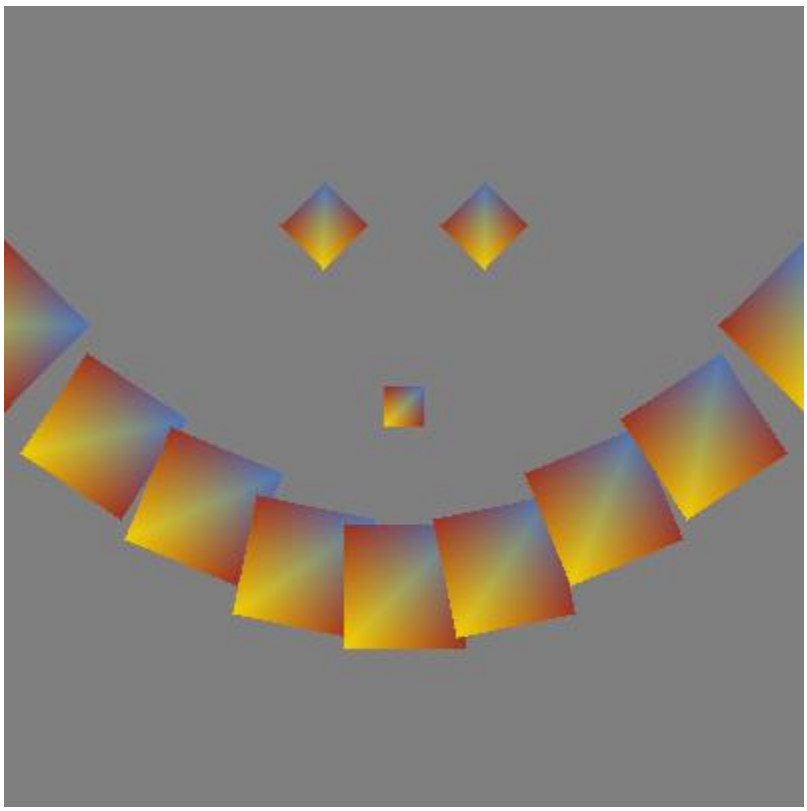
Question 6:

Draw rainbow_square

Nose: scale – 0.1, 0.1 translation - 0.0, 0.0 rotate – 0.0

Eyes: scale – 0.15, 0.15 translation – 0.2, 0.45 and -0.2, 0.45 rotate – pi/4

Mouth: scale – 0.3, 0.3 translation – x = -1.0, -0.75, -0.5, -0.25, 0.0, 0.25, 0.5, 0.25, 1.0 y = 0.2, -0.075, -0.25, -0.40, -0.4, -0.25, -0.075, 0.2 rotate - -pi/4, -3pi/16, -pi/8, -pi/16, 0, pi/16, pi/8, 3pi/16, pi/4



5 Make an Image!

I was inspired by the Sierpinski triangle though not an infinitely recursive one. I created a .tri file that started with the triangle from rainbow_square.tri. I then added to the .tri file the lines that support scale, translation and rotation. The make_an_image.tri uses all three of these to create the image below.