Our project involved visualizing the fractals generated from systems of contraction mappings known as an iterated function system (IFS). If we iterate the family of functions starting with an arbitrary set, we get a sequence of sets that converges to a unique fixed point set, which is the fractal associated with the IFS. An example of this is the Sierpinski triangle, which turns out to be the fractal associated with three maps, the maps being contractions by a factor of two towards three points in the plane.

Given a collection of functions $f_1, f_2, \ldots, f_k$, start with any point. Then iteratively select one of the functions at random and apply it to the point. This will give a sequence of points approaching the fixed point set. The selection of functions should be appropriately weighted to produce a quality image. If the function $f_i$ has an associated contraction factor $r_i$, then it needs to be chosen with weight $r_i^s$ such that $r_1^s + \cdots + r_k^s = 1$. We consider only affine maps and represent them as triangles. There is always an implicit big triangle being mapped to smaller triangles; the maps being the unique affine maps that take the big triangle to the respective smaller triangles.

1 Files

Here is a list of important files in the project.

1.1 MainTriangleDemo.nb

The section “init” contains the appropriate initialization definitions. The section “constrained” initially draws the Sierpinski triangle and allows the user to move certain vertices; the remaining are determined so that all triangles are equilateral. Here, the number of triangles is $3^{n-1}$ and $n$ is defined at the top of the cell. The next section “free” deals with equilateral triangles in general position. The number of triangles is $n$. The final section deals with triangles that are not necessarily equilateral.

1.2 FourTriangleDemo2.cdf

This is a version of the final section of the previous notebook, for $n = 4$, suitable for browser embedding.

1.3 TriangleMorpher.nb

This notebook creates an animated gif of the deformation described in “qcdim-notes.pdf”. The first cell must be run with $NN$ set to the desired value of $N$. The third cell computed the frames, the fourth cell can be used to export the frames into a gif.

1.4 Sierpinski2Dragon.nb

This notebook is used to animate linear interpolations between IFS’s. The first cell compiles the interpolation so that it will run faster. It is currently set up to handle three functions; if you would like to change the number of functions, add the appropriate cases to the “Which” statement and modify the list of numbers in the “RandomChoice” function in third cell.
1.5 muMath.exe

This program allows the user to draw the IFS fractal for the system of functions defined on $C$. The first argument is the list of functions in the IFS. The second argument is of the form 
\{\text{var}, \text{LowerLeftCorner}, \text{UpperRightCorner}\}. For example, the command

\[
\text{IFSPlot2D}\left[\{(1+I)/2 \ z, 1 - (1-I)/2 \ z\}, \{z, -0.5 - 0.5 \ I, 1.5 + 1.5 \ I\}\right]
\]

will draw a dragon curve.