Fractals: Visualization with Variation in Affine Transformation

By: Sergio Castanon-Diaz & Ryan Furukawa
Math 2270 - Linear Algebra

What are Fractals?

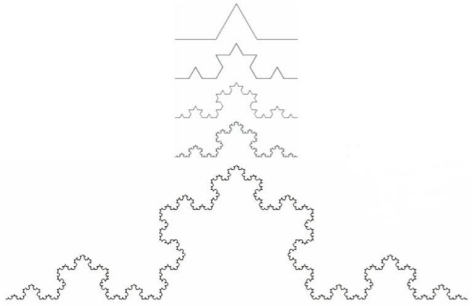
Iterated Function System (IFS) Fractals

For our project, we focused mainly on the iterated function system category of fractals because of their exact self-similarity quality. For example, the tree fractal that we generate looks identical at any scale. We take a look at one of the branches and it looks exactly like the bigger picture. The same occurs with the modified Sierpinski triangle that we generate. If we were to zoom into the fractal's smaller pieces, we would see the same complexity as in the larger view of the same fractal.

The fractals in our case are generated by translating, rotating, and scaling the original shape that we start out with. This way, there is a function that can be applied in order to make the fractals in a recursive manner where the size of the fractal copies keeps decreasing. Because of this method of creation/generation, we produce exact self-similar fractals in the end result.

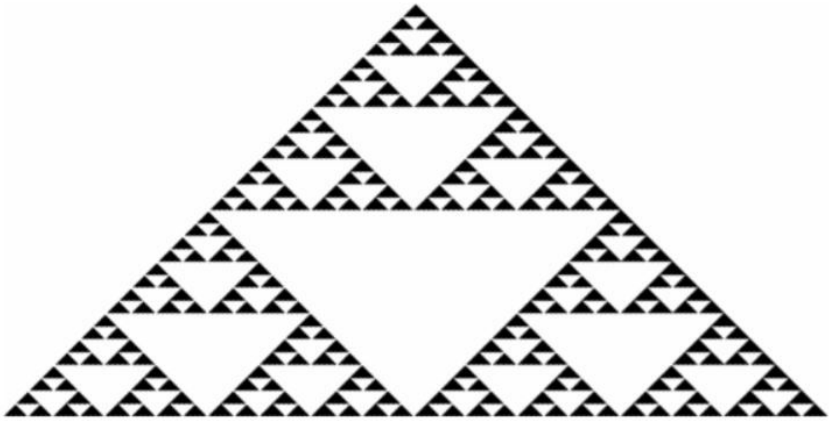
A Few Examples of IFS Fractals

Koch Curve - The initial shape is the first image, then all four of the straight line segments are replaced by the initial image so that there are now four "bottomless triangles" and this is repeated until we reach the bottom fractal where if we zoom in, we will see the same initial shape in all of the smaller line segments.



Picture sourced from [2]

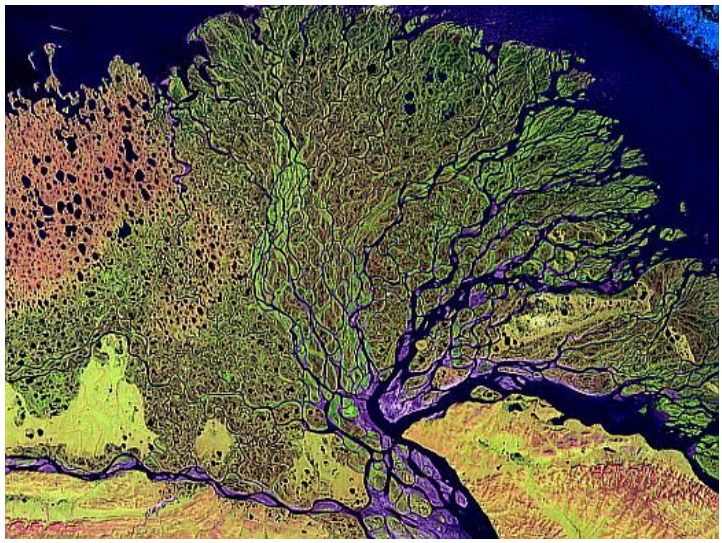
Sierpinski Triangle - The Sierpinski Triangle fractal can be made by taking the initial triangle in the middle and then translating a scaled triangle to each of the three sides of the original. This process can be repeated for each of the new triangles and we end up with the image like below.



Picture sourced from [2]

Fractals in Nature

Apart from being able to create fractals ourselves, fractals also occur naturally as well. A pair of examples of fractals in nature are river deltas and, of course, real trees. Below are two pictures where we can see fractals in nature and even though the fractals are not exactly self similar, they do resemble some of the screengrabs that we were able to produce with our fractal generator.



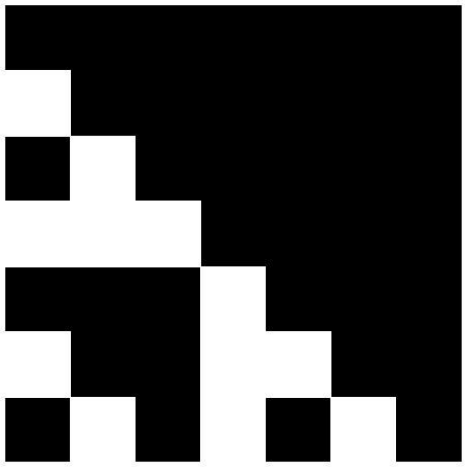
Left image of tree sourced from [3] and right image of river deltas sourced from [4]

Creating Fractals in Java

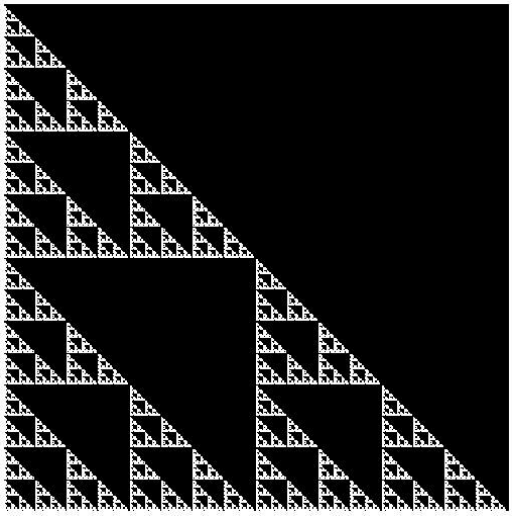
For our Fractal Generator, we went with Java since we were fairly familiar with drawing shapes and doing transformations on them with Java libraries. Plus, there is an AffineTransformation class that makes things simple to understand since the library gives examples of the matrices used to perform the transformations but there are also standard scale, translate, and rotate methods as well. When actually doing the project, we found it easier to just use the standard methods of transformations (especially when rotating) because then the next values to be used could be calculated and passed into the next iteration of recursion.

Now below are some examples of the fractals that can be created with the generator:

Squares:



This one was made with three functions where the level of recursion was 4. If we stare at it for long enough, we can begin to see a pattern like the one above for the Sierpinski Triangle. The three black squares are like the three scaled triangles that are translated from the original triangle. In our case, we have an original square and the original “square” looks more like a military spy aircraft but that is only because the level of recursion was not very deep so the final result is not detailed. In the next image, we can see the result of a higher depth of recursion.



Now, with a higher level of recursion (this time the recursion level was 9) we can see that the result resembles a Sierpinski triangle even though our initial shape was a square. So, with enough recursion, basically any initial shape will result in a similar Sierpinski triangle.

Trees:

The trees and triangles below follow the pattern:

$$f(x) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} e \\ f \end{bmatrix}$$

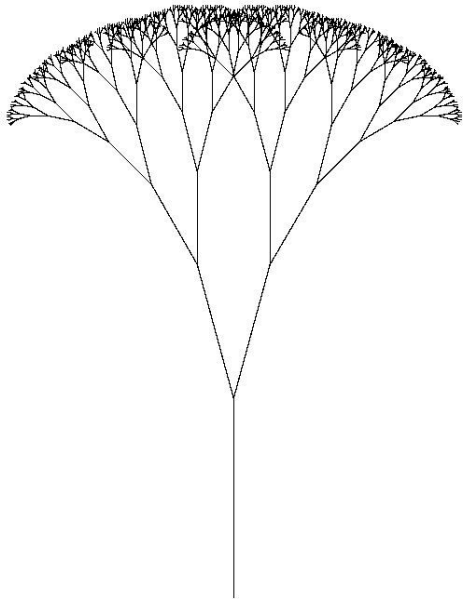
Where the e and f variables correspond to a translation and then a, b, c, d correspond to the values of a transformation matrix. X is also a 2x1 matrix like the translation and contains the x and y of the current iteration.

The equations for the trees consist of a rotation and scaling as well as a small translation so their transformation matrices follow the pattern below:

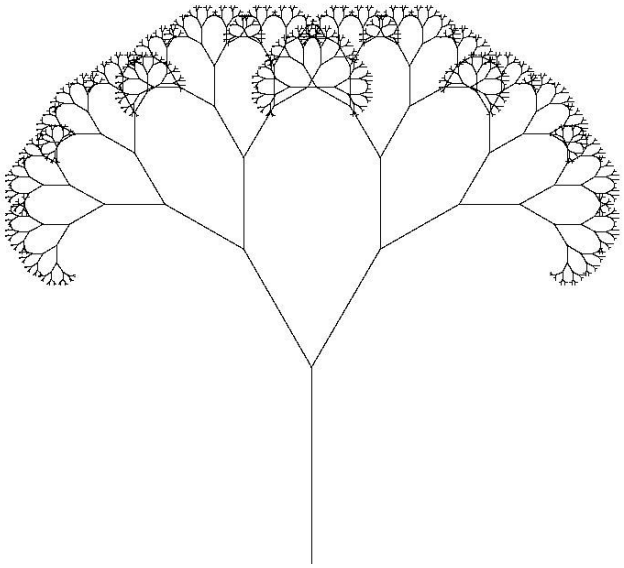
$$L = \begin{bmatrix} r\cos(\text{angle}) & -r\sin(\text{angle}) \\ r\sin(\text{angle}) & r\cos(\text{angle}) \end{bmatrix}$$

$$R = \begin{bmatrix} r\cos(\text{angle}) & r\sin(\text{angle}) \\ -r\sin(\text{angle}) & r\cos(\text{angle}) \end{bmatrix}$$

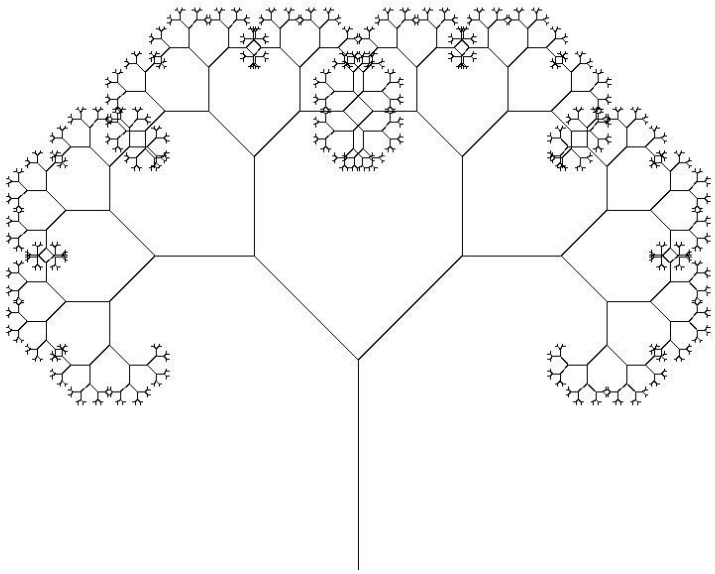
For all of the trees, the r or scale factor is ⅔. For the project, we focused on changing the angle that the branches were rotated at starting with 30 degrees and ending with 120 degrees. The level of recursion for each of the trees was 16, so there are 16 iterations in each of the trees below even though some of the details get lost in the later iterations. Now on to some screenshot examples of the tree fractal results.



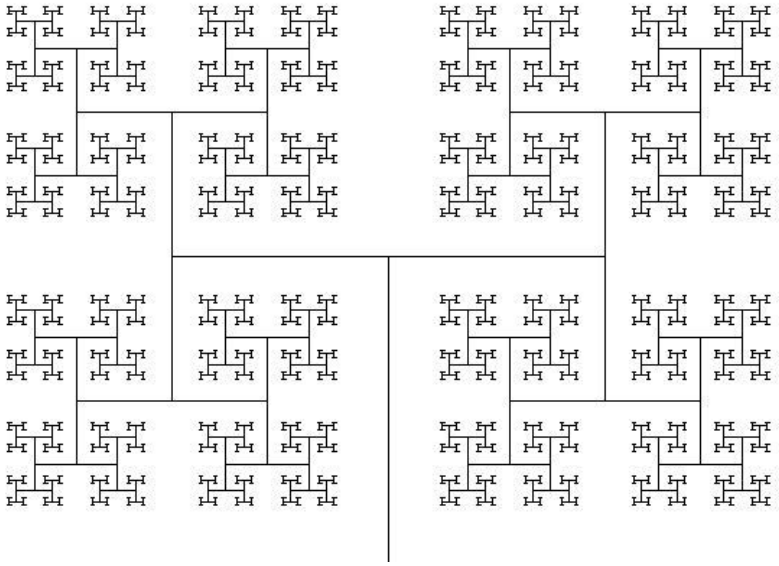
This one was made with each of the branches being split by 15 degrees and we can see that it is the tallest out of all of the other trees that we took screenshots of. This makes sense since the smaller that the angle that we rotate by is, the less that the new branches deviate from the previous branch.



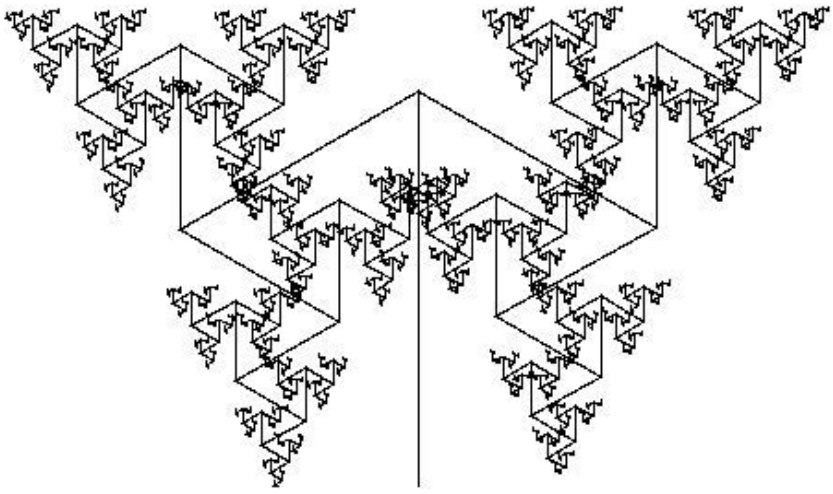
The branches on this tree are split by 30 degrees and we begin to see more of a spread in the final branches as well as more overlapping in each of the branches. The overall shape of the edges of branches is also not quite as round as in the tree in the image from above, instead, we start to see some segments that look straight, especially up top.



These branches are split by 45 degrees and this is where the fractals start to look more interesting. The edges on this one look like the top half of an octagon (but only because the two sides overlap, otherwise we would see half of a hexagon in both of the top edges) and then the inner line segments resemble some pixel hearts, plus we can see that the deeper we focus/zoom into the fractal, the same patterns still occur.



The images above and below are the fractal results that are the most interesting and pleasing to the eye as the “trees” begin to deviate from the trees that we are used to seeing in nature. In the image above, the fractal’s branches are split by 90 degrees and that leads to a unique look where all of the lines are either parallel or perpendicular to each other. Plus, this fractal makes it simple to see that the iterations of the original shape are just scaled and translated to form the smaller fractals.



Finally, these branches are split by 120 degrees and compared to all of the other trees, this one actually has more of a triangle shape to it. Also, the image is a lot more dense than the others and that is mostly because the degrees are now hyperextending the new branches inward instead of outward like the others.

Bibliography

1. Byrd C. ‘Fractals in Nature: An introduction to IFS and L-System Fractals’ https://web.cs.wpi.edu/~matt/courses/cs563/talks/cbyrd/pres1.html

2. Supina P. (2005-2006), ‘Visualization of Fractal Sets in Multi-Dimensional Spaces’, http://www.math.utah.edu/~gustafso/s2019/2270/pdf/fractals-petr-supina-2005-bp_vis.pdf

3.https://i.pinimg.com/originals/05/1a/1b/051a1bdd32099276dcac1fc760d2a8aa.jpg

4.https://media.science360.gov/files/pic-day/fd376fd2-e5a5-4098-a999-759301141c82-largeImage.jpg