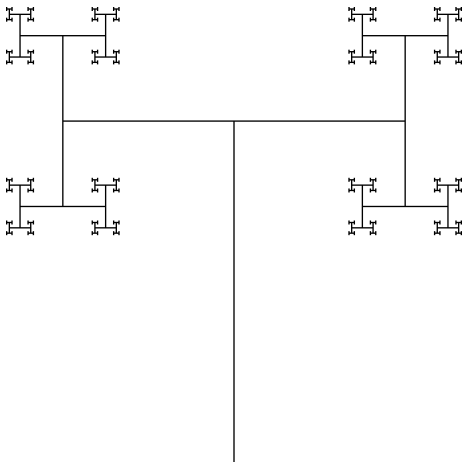
The Dimension of Mutually Recursive Fractals

Aresh Pourkavoos

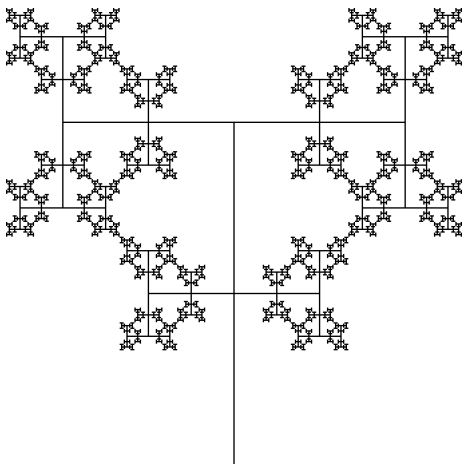
December 30, 2020

Once upon a time, I was playing with a Processing program I had written to draw binary trees.



A binary tree drawn by the program.

This tree of height h is drawn by drawing a trunk of height h (a straight line) and two smaller trees of height $\frac{h}{2}$ coming out at right angles. However, this tree looks very sparse, more like a dead tree or one that has lost its leaves for the winter. In the interest of lushness, I decided to fill in some of the empty space by adding branches to the trunk: two of them, coming out at right angles from the center of the trunk and with height $\frac{h}{4}$ (to be smaller than the main branches). The program drew the following tree.

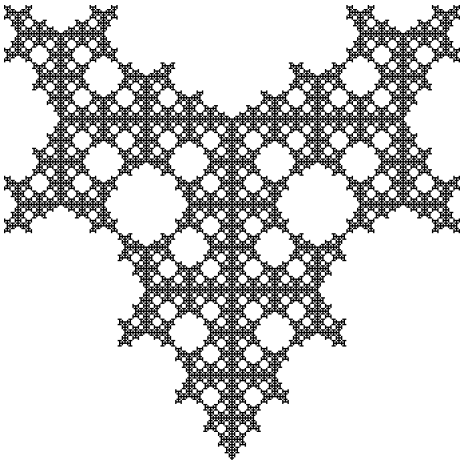


This looks a bit more interesting and a bit less dead. The tree's "leaves" look like they form long chains, but they're disconnected if you look closely enough. The large and small branches seem to just barely touch without overlapping.

Most importantly, there was still plenty of empty space in the form of long segments with no branches. I wanted to add more to the trunk, but the code was getting longer and it was becoming more difficult to calculate where to place each new branch. Then, I had the idea of using **mutual recursion**.

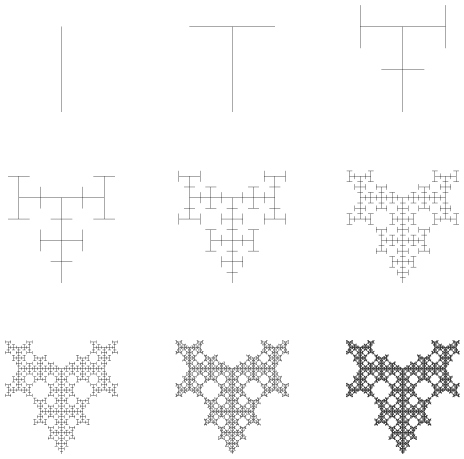
The function to draw the trees above is recursive in the normal sense, since it calls itself to create the smaller branches. On the other hand, **mutually recursive functions call each other.** Instead of a straight line, I wrote a new function to draw the trunk, which draws the two smaller branches (trees) as well as **two trunks** of height $\frac{h}{2}$ stacked on top of each other. That way, the half-trunks would themselves draw new, even smaller branches, as well as even smaller quarter-trunks, etc. In the limit, each trunk would spawn an infinite number of branches, each new round smaller than the last, eventually filling all of the empty space on the trunk of the original tree!

The program spat out this.

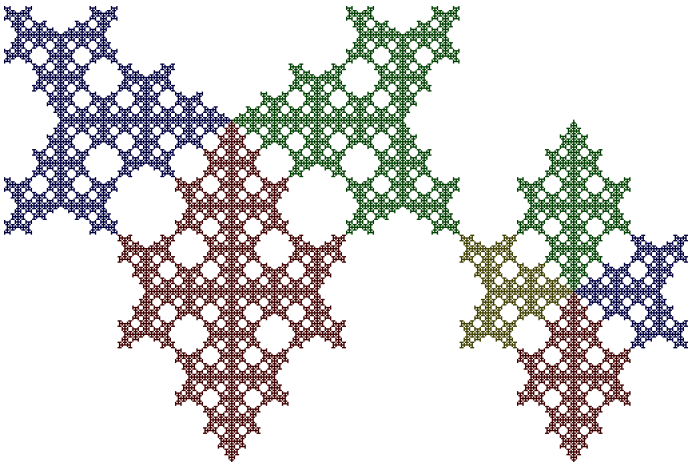


The mutually recursive tree.

The new shape jumped out at me. The tree now has a clearly-defined inside and outside, there no longer seems to be a distinction between tree and trunk if you zoom in enough, there are symmetrical shapes in its negative space that remind me of lace doilies, and the entire thing looks like the skull of a dog or similar animal.



The first few iterations.



The tree (left) and its trunk (right), color-coded to show the elements of each.

Needless to say, I was intrigued. I had never seen this shape before, and one question that came to mind was "What is its fractal dimension?" If you aren't familiar with the idea of fractal dimension, this video offers a great explanation with nice visuals. The first nine minutes or so are all you need for the next section (since this tree is perfectly self-similar), but I would highly recommend the entire video anyway.

I found the fractal dimension of this tree in two ways. The first is a fairly direct application of the technique in the video and creates a tree solely out of scaled-down copies of itself. The second, which I find more elegant, takes full advantage of the tree's mutually recursive nature with the help of some linear algebra. This second method generalizes well to other fractals defined in a mutually recursice way. But, just to build suspense, let's cover my original method first.

By the (simplified) definition of the fractal dimension D, when we scale the tree by some factor s, its "mass" is scaled by the factor s^D. Since the tree is built (at least partially) out of copies half its size, setting $s = \frac{1}{2}$ seems like a reasonable choice. Let our original tree have mass 1, so its two largest branches each have mass $(\frac{1}{2})^D$.

But, what's the mass of the trunk?

First, rewrite the tree and trunk of height h in terms of trees and trunks of height $h/2$. Within the trunk of height h, 1 of the trunks of height $h/2$ and the 2 trees of height $h/4$ make up a tree of height $h/2$. Thus a trunk of height h is 1 trunk of height $h/2$ and 1 tree of height $h/2$. Within the tree of height h, the trunk of height h can be rewritten with this new definition. Thus a tree of height h is 1 trunk of height $h/2$ and 3 trees of height $h/2$.

Let a be the "mass" (measure?) of a trunk with height $h/2$, and let b be the mass of a tree with height $h/2$. Then the mass of a trunk with height h is $a + b$, and the mass of a tree with height h is $a + 3b$. Thus the (linear) transformation which maps the vector $\begin{bmatrix} a \\ b \end{bmatrix}$ to the vector $\begin{bmatrix} a+b \\ a+3b \end{bmatrix}$ scales the heights of a trunk and a tree by 2. It can be represented by the matrix $\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$. Assume that trunks and trees have the same Hausdorff dimension d. (Proving this may be nontrivial, but the interiors of trees and trunks are visually similar.) Thus scaling a tree's (or a trunk's) height by a factor of 2 scales its mass by a factor of 2^d. Call this quantity λ. Since the matrix $\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$

Use the writeidea command to edit.