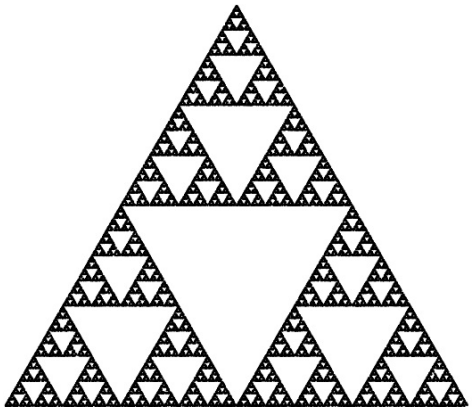
Fractal Trees

Gandhi Games

Table of Contents

Fractals

A fractal is a pattern that repeats at different scales. We call these shapes “self-similar”. While they have only been known as fractals since the 1970s, they were first described by Leonardo da Vinci in the 15th century.



A self-similar/fractal shape.

Nature is full of fractal patterns including trees, lightning bolts, and river networks. Spirals such as hurricanes and even galaxies are also considered fractals.

Fractal Trees

A tree is a type of an approximately self-similar fractal, where smaller parts of the tree look generally similar to the larger tree. A tree is formed by repeating a simple process, which is also the basic principle that is used when generating fractals programmatically. While the fractals generated can appear complex and detailed, they are created by repeating a number of simple steps.

The algorithm for growing a tree is roughly like so:

1. The tree sprouts.
2. The sprout eventually splits into branches.
3. These branches themselves split into further branches.

At each step of this process it is as if two new smaller trees emerge. These smaller trees can be conceptualized as the trunks of a new generation of trees. This repetition of branching that forms the tree is also the cause for trees approximate self-similarity. In nature this process eventually stops and the end products are no longer fractals.

Fractal Trees in Nature.

There are a number of different methods to generate fractal trees programmatically, including the Lindenmayer system and Space Colonization.

Lindenmayer Systems

Computers are important tools in the study of the structural patterns in natural and computer generated organisms. Lindenmayer systems, shortened to L-systems, (introduced in 1968) were one of the first occurrences of the use of computational power to study these patterns.

An L-system creates sets of strings based on a rule set. The system starts with an Axiom (or seed), and then rules are recursively applied to the string to produce an output string or sentence. The output sentence can then be fed into other systems to produce graphical output.

For example:

Axiom: AB
Rules: A -> AB, B -> A

Gen 1: AB (Initial Axiom)
Gen 2: ABA (A converted into AB and B into A)
Gen 3: ABAAB
Gen 4: ABAABABA
Gen 5: ABAABABAABAAB

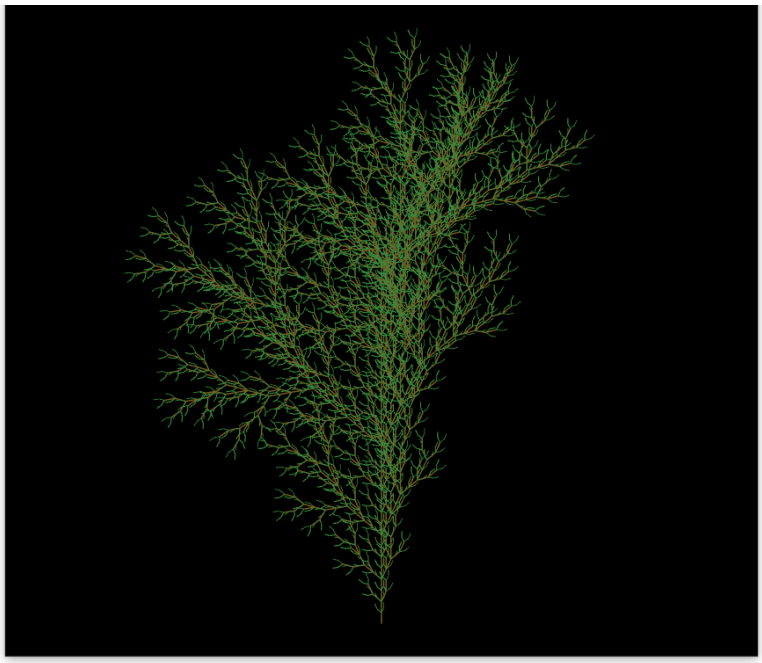
Generation one starts with the axiom. In subsequent generations any instance of ‘A’ is replaced with ‘AB’ and ‘B’ with ‘A’.

By giving each character a specific action and implementing turtle graphics (https://en.wikipedia.org/wiki/Turtle_graphics) we can generate trees.

The system uses a version of the d0L-system Grammar Description Language. This language provides rules that can be applied to each character generated by the L-System.

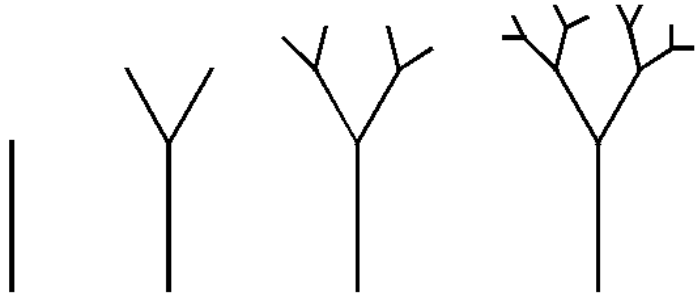
Character	Rule
C0, C1… Cn	Change all subsequent drawn lines to colour n.
F	Move forward in current direction and draw line.
G	Move forward in current direction without drawing a line.
-	Rotate direction left by n degrees.
+	Rotate direction right by n degrees.
[	Store current state.
]	Restore saved state.
!	Reverses the meaning of left and right.
|	Turn 180 degrees.

Using these actions and a specific Axiom and rule set, you can create the tree shown in the image below.



Axiom: FX
Rules: F -> C0FF-[C1-F+F]+[C2+F-F], X -> C0FF+[C1+F]+[C3-F]
Generations: 5

Each generation produces a tree more complex than the last. An example of the first four generations of a similar rule set is shown below.



Further example rule sets used can be found online, for example:
http://www.kevs3d.co.uk/dev/lsystems.

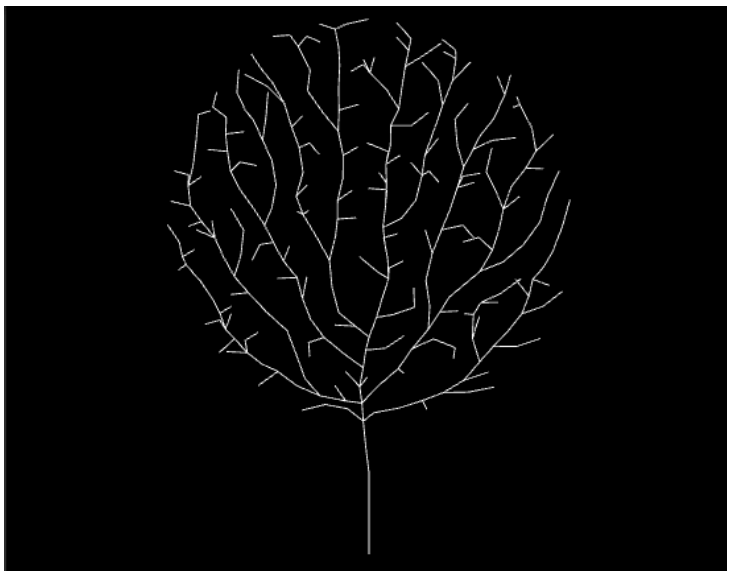
Space Colonization

Space Colonization is another method of generating trees. You start by defining the leaves. These leaves act as a target for branches to grow towards. They can be spawned randomly or hand placed. The tree's trunk is then grown until it is within a specified distance of a leaf. From this point, the tree will grow towards the leaves.

For each branch end an attraction vector towards leaves within distance is added. This vector is then used to spawn new branches and the process continues. Whenever a segment end is too close to a leaf, the leaf is removed. As the branches grow towards the leaves most (if not all) leaves will be removed at the end of the tree generation process.

Using this process, branches naturally avoid each other; each branch appears to have developed as the result of seeking sunlight. This same method can also be used to generate the roots of the tree.

You can create vastly different trees by adjusting the minimum and maximum distance seek distance to leaves, and the width of branches.



An example of a tree generated using Space Colonization.
A random circular spread of leaves was placed prior to generation.