L-Systems - Fractal Trees

Pranav Phadke

April 21, 2022

Abstract

This paper explores the mathematical beauty behind what is known as L-Systems (Lindenmayer Systems). We will begin by explaining what exactly an L system is. We will discuss the history and concept behind their application. Furthermore, we will then discuss their self-symmetrical fractal nature and their ability to represent plant growth. Lastly, we will discuss the logic and math behind a JavaScript code that visualizes different L-Systems. All code with its documentation can be found here:
https://github.com/pranavphadke1/L-Systems-Fractal-Trees

1 Aristid Lindenmayer

L-systems were first introduced in 1968 by Hungarian theoretical biologist and botanist Aristid Lindenmayer. Lindenmayer was working with yeast and fungi and was studying the growth of patterns of different types of bacteria. At first, his studies led to a formal description of the growth of simple multicellular organisms, specifically the growth of algae. However, this systems was later extended to describe the growth of plants - which is what we will focus on.

2 L-System

An L-System is essentially a string rewriting system. It is a text-based recursive algorithm that is in the class of formal grammars. The idea behind an

L-System is to generate a never-ending sentence through the combination of a starting element, referred to as an axiom, and variable replacement rules. An L-System can be described by the tuple $G = <V, w, P>$ where:

- V = the alphabet (a set of elements) containing both elements that can be replaced (called variables) and those that can not (called terminals).
- w = the starting element (the axiom)
- P = the set of element replacement rules

Let's consider a simple L-System that only contains variables and no terminals:

- Axiom: "A"
- Rules:
- $A \to AB$
- $B \to A$

Using this L-System, we can result in the following substitutions:

- Step 0 (Start) : A
- Step 1 : AB
- Step 2 : ABA
- Step 3 : ABAAB
- Step 4 : ABAABABA

This L-System is fairly simple, but it is clear how the replacement rules can be used to recursively substitute variables never-endlessly.

Let us also look at an L-System that contains terminals.

- Axiom: "F"
- Rule:
- $F \to FF + [+F - F - F] - [-F + F + F]$

Following this L-System, we get:

- Step 0 (Start) : F
- Step 1 : FF+[+F-F-F]-[-F+F+F]
- Step 2 : FF+[+F-F-F]-[-F+F+F]FF+[+F-F-F]-[-F+F+F]+[+FF+[+F-F-F]-[-F+F+F]-FF+[+F-F-F]-[-F+F+F]-FF+[+F-F-F]-[-F+F+F]]-[-FF+[+F-F-F]-[-F+F+F]+FF+[+F-F-F]-[-F+F+F]+FF+[+F-F-F]-[-F+F+F]]

In this L-System, we can see that there are symbols (+,-,[,]) that do not get replaced. We can also see that the resulting ”sentence” grows extremely quickly. This specific format of an L-System is one that we will discuss. When translated into images, the first 5 steps of the L-System can be represented in the figures below. Due to its tree-like shape, I call the result of this L-System a Bent Tree.

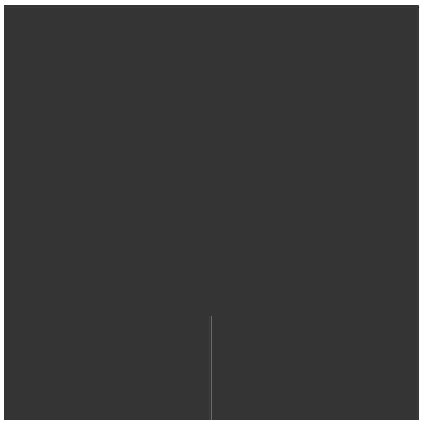

Figure 1: Step 1

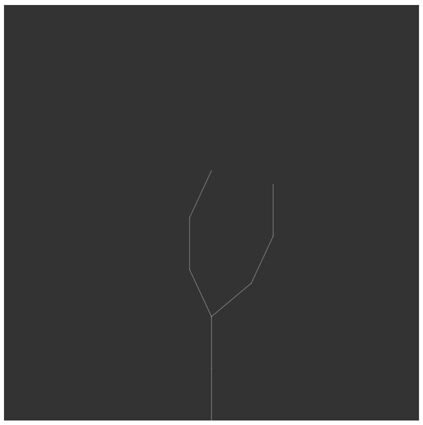

Figure 2: Step 2

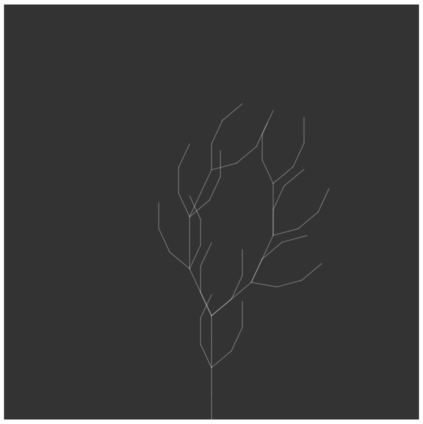
Figure 3: Step 3

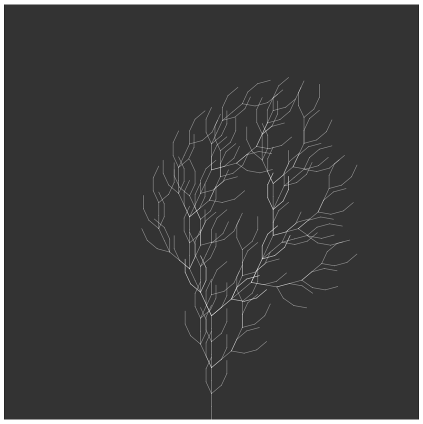
Figure 4: Step 4

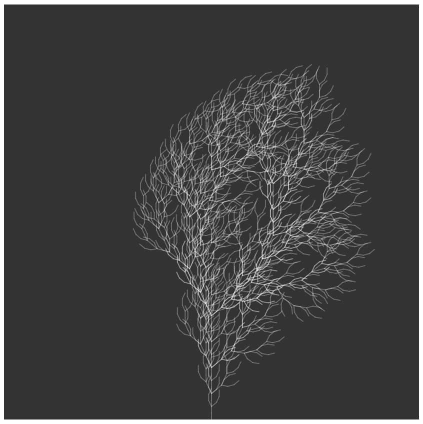
Figure 5: Step 5

3 Turtle Graphics — Turtle Function

All of the graphical representations of various L-Systems we will describe have been encoded through what is known as Turtle graphics. In the JavaScript code, there is a function called Turtle that essentially draws a graphic by creating several lines and moving across a canvas. This concept is based off an old educational based programming language called Logo.

What this turtle function allows us to do is keep track of a point on the canvas; draw straight lines from that specific point; rotate right or left some amount of degrees; save the current point (related to pushing onto a stack); and returning to the last previously saved point (related to popping from a stack).

4 Text To Graphic Encoding

The L-System we described above is one which has 4 terminals and the variable F. These 5 symbols are used to encode a graphical representation of the L-System. We use the Turtle function previously described and encode as so:

- F means to draw a straight line
- + means to rotate right
- - means to rotate left
- [means to save the current location
-] means to return to the previously saved location

5 Self-symmetry and Fractals

The recursive nature of the rules of the L-System leads to self-similarity and a fractal-like pattern appears after each substitution. As seen in https://www.youtube.com/watch?v=0eXg4B1feOYt=561sab$_{c}hannel = Confreaks$, it is possible to represent common fractals like the Koch snowflake and Sierpinski's Triangle, but the provided JavaScript code focuses on tree-like fractals representing plant growth.

6 Math behind the JavaScript Code

The code is written in the JavaScript extension for the Processing programming language. It is essentially JavaScript with additional prebuilt functions from the Processing library that allow us to create visualizations.

6.1 Turtle Function

This function does the actual visualizations of the current "sentence". We already discussed Turtle on an abstract level above. In terms of the actual code however, there are some variables I left out in the previous discussion. At the top of the code, there are different L-Systems commented out. You must uncomment the L-System you wish to use. For each L-System, there are different angle values that are necessary for a pretty graphical representation. The angle values are in degrees, however the built in rotate() function requires radian values. The angle values are converted to their corresponding radian value through the radian() function used in the setup function. Additionally, the rules for each L-System are encoded a specific way which we will talk about in the next section.

6.2 How Rules are Stored

Rules are represented by JSON objects (JavaScript Object Notation) where the value of the first attribute "a" represents the symbol that will be replaced and the value of the second attribute "b" represents the substitution. Thus the rule A $\rightarrow$ AB is represented by the JSON object $\{a : A, b : AB\}$. All of these rules are stored in an array. This is useful for the generate function which we will discuss next.

6.3 Generate Function

This function is what does the actual substitutions at each level. It essentially goes through every symbol in the current sentence, applies substitutions based on the array of rules, discussed in the previous section, and then saves the new sentence. Once the new sentence is saved, this function calls the previously discussed Turtle() function to render this new level.

7 More L-System Fractal Trees

Here are some more L-Systems with their rules and resulting graphics.

Arrow Weed. Figure 6

- Axiom: "X"

- Rules:
- $F \rightarrow FF$
- $X \rightarrow F[+X][-X]FX$

Fuzzy Weed. Figure 7

- Axiom: ”X”
- Rules:
- $F \rightarrow FF$
- $X \rightarrow F-[[X]+X]+F[+FX]-X$

Tall Seaweed. Figure 8

- Axiom: ”F”
- Rules:
- $F \rightarrow F[+F]F[-F]F$

Twiggy Weed. Figure 9

- Axiom: ”X”
- Rules:
- $F \rightarrow FF$
- $X \rightarrow F[-X]F[-X]+X$

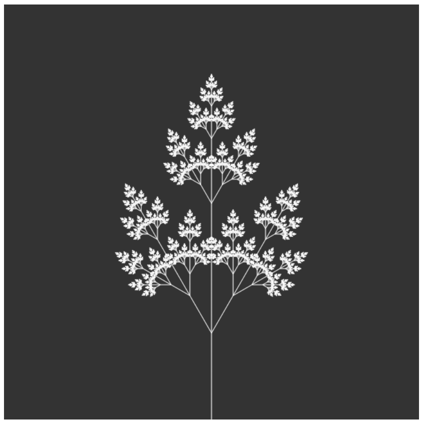
Figure 6: Arrow Weed

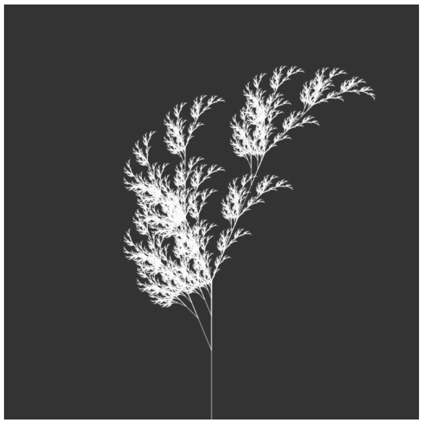
Figure 7: Fuzzy Weed

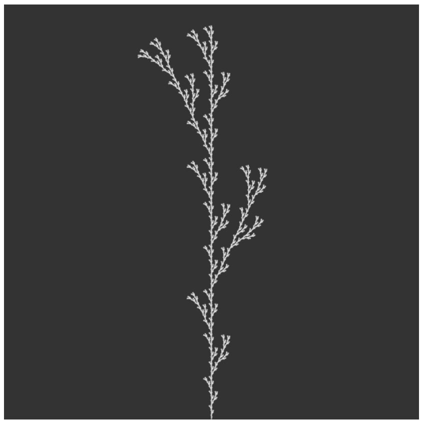
Figure 8: Tall Seaweed

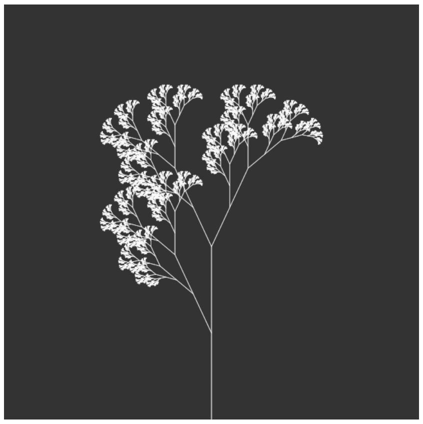
Figure 9: TwiggyWeed

8 References

1. https://github.com/pranavphadke1/L-Systems-Fractal-Trees
2. TheCodingTrain YouTube Channel
3. https://www.youtube.com/watch?v=0eXg4B1feOYt=561sab$_channel = Confreaks$