Introduction to L-Systems and Fractals

Lesson Worksheet

DEFINITION: An L-System is a parallel re-writing system, commonly used to construct structures resembling fractals. Essentially, this means an L-System consists of a set of *objects and *relations to modify the structure based on iterations.

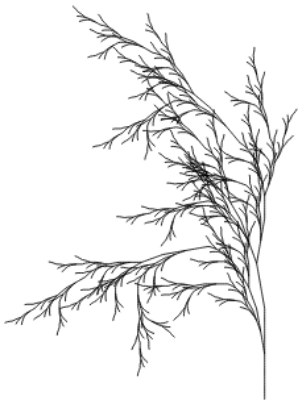
L-Systems are recursive; each iteration is defined by a previous version of itself. Such a pattern creates what we call a self-similar structure, a structure defined as one exactly resemblant of, or approximately similar to the structure's prior version.

We see self-similar structures in nature everywhere! There is even self-similarity in coastlines! However, there are many more examples, especially in plants.

Activity #1

What are some examples of self-similar objects in nature? Think of one or two, google is allowed, but see what you can come up with on your own first!

When you think of one, try to draw it out!



Example: A Tree!

Objects: anything contained in the L-System's set. For our purposes, these will usually be variables and maybe a few symbols.

We construct an L-System by using three parameters: the alphabet, axiom, and production rules/relations. This is typically written as:

G = (V, ω, P)

V = the "alphabet," which is the set of all changeable elements (variables) and immutable ones (constants)

Ex: {A, B}

ω = the "axiom," which is the value you start the structure with. In most L-Systems, the axiom is iteration 0. For simplicity, we'll be focusing on structures using a single-variable axiom, but axioms can work with multiple variables

Ex: ω = A

P = the "production rules" or "relations," which define how the variables will be replaced in each iteration. Production rules typically consist of a "two-part-system" with a *predecessor and a *successor, but we won't worry about those quite yet. In production rules, the "=" operator functions to re-define a variable

Ex: (A = AB), (B = A)

When a computer reads an "A" it will replace the value with an "AB" after one iteration.

EXAMPLE RESULT:

Iteration 0: A

Iteration 1: AB

Iteration 2: ABA

Iteration 3: ABAAB

Iteration 4: ABAABABA

Iteration 5: ABAABABAABAAB

Activity #2

We're going to create our own L-System strings! Have fun with this… nothing is too chaotic

First, define your *alphabet*:

V =

Next, define your *axiom*:

ω =

Next, define your *production rules*:

P =

Iteration 0:

Iteration 1:

Iteration 2:

Iteration 3:

Iteration 4:

Iteration 5:

The processor and successor parts are essentially your "a_{n-1}" and "a_n," you use the previous string to define the next, making the recursive part of L-Systems.

Constants: these are typically denoted as symbols in the alphabet, ex: "]" and "[". Constants have an extra set of rules assigned to them that helps computers render an image from the strings we created… so essentially… FRACTALS!

Variables will also often be assigned extra rules when rendering an image.

Activity #3 (CHALLENGE)

V = {1, 0, [,], } ω = 0 P: (1 → 11), (0 → 1[0]0)

We're making the "binary tree!" This is a great example of how L-Systems look like natural growth patterns.

Extra Rules:

0 = draw a line segment with a "leaf" tip at the end (pretty much just a segment where another iteration will continue off from)

1 = draw a line segment

[= a push (saves position and angle until]) and rotate left 45°

] = a pop (returns to position prior to [ ) and rotate right 45°