# CMSC 425: Lecture 13
# Procedural Generation: L-Systems

**Reading:** This material comes from Chapter 1 of *The Algorithmic Beauty of Plants*, by P. Prunsinkiewicz and A. Lindenmayer, 2004. It can be accessed online from http://algorithmicbotany.org/papers/.

**L-Systems:** Today, we will consider the question issue of how to automatically generate "tree-like" objects, which are characterized by a process of growth and branching. The standard approach is through a structure called an *L-system.* L-systems, short for *Lindenmayer-systems*, were first proposed by a biologist Aristid Lindenmayer in 1968, as a mechanism for defining plant development.

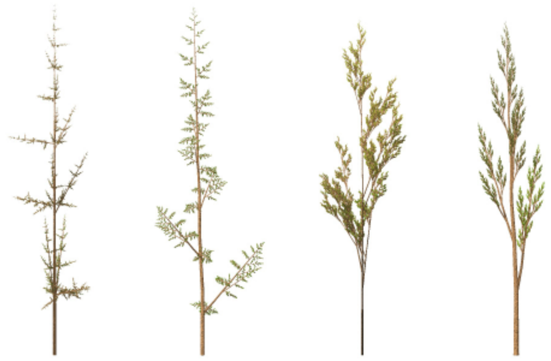


Fig. 1: Examples of simple plant models generated by L-systems.

If you have taken a course in formal language theory, the concept of an L-system is very similar to the concept of a context-free grammar. We start with an alphabet, $V$ which is a finite set of characters, called *symbols* or *variables.* There is one special symbol (or generally a string) $\omega \in V$, called the *start symbol* (or *start string*). Finally, there is a finite set of *production rules.* Each production replaces a single variable with a string (or zero or more) symbols (which may be variables or constants). Such a rule is expressed in the following form:

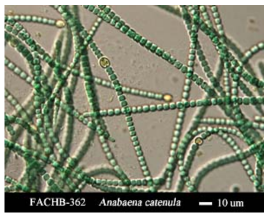
$$\langle\text{variable}\rangle \;\rightarrow\; \langle\text{string}\rangle.$$

Letting $V$ denote the variables, $\omega$ denote the start symbol/string, and $P$ denote the production rules, an L-system is formally defined by the triple $(V, \omega, P)$.

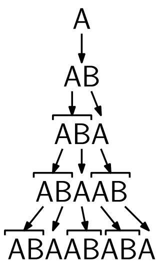
Symbols are categorized in two types: *variables* are symbols that appear on the left-hand sides of production rules and can be replaced, and *constants* (or *terminals*), which cannot be replaced. An L-system is said to be *deterministic* if for each variable, there is a single rule having this variable on its left side.

To get a better grasp on this, let us consider a simple example, developed by Lindenmayer himself to describe the growth of the *Anabaena catenula algae* (see Fig. 2(a)). The variables are $V = \{\mathsf{A}, \mathsf{B}\}$, there are no constants, the start symbol is $\omega = \mathsf{A}$, and the rules are:

$$P \;=\; \{\mathsf{A} \rightarrow \mathsf{AB}; \quad \mathsf{B} \rightarrow \mathsf{A}\}$$



(a)



(b)

Fig. 2: L-system modeling the growth of the Anabaena catenula algae.

An L-system works as follows. Starting with the start symbol, we repeatedly replace each variable with a compatible rule. In this case, each occurrence of A is mapped to AB and each occurrence of B is mapped to A. This is repeated for some desired number of levels (see Fig. 2(b)).

As an aside, if you count the number of symbols in each of the strings, you'll observe the sequence $\langle 1, 2, 3, 5, 8, 13, 21, \ldots \rangle$. Is this sequence familiar? This is the famous Fibonacci sequence, which has been observed to arise in the growth patterns of many organisms.

**Using L-Systems to Generate Shapes:** So what does this have to do with shape generation? The idea is to let each symbol represent some sort of drawing command (e.g., draw a line segment, turn at a specified angle, etc.) This is sometimes referred to as *turtle geometry* (I guess this is because each command is thought of as being relayed to a turtle who carries out the drawing process).

The system is defined by two parameters, a *step size* $d$ and a *angle increment* $\delta$. The *state* of the turtle is defined by a triple $(x, y, \alpha)$, where $(x, y)$ is the turtle's current position and $\alpha$ is its *heading*, the direction it is currently facing (see Fig. 3(a)). Define the following commands:

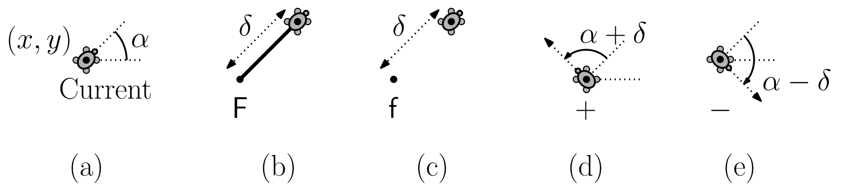


Fig. 3: Turtle-geometry operations.

- "F": Move forward by a step of length $d$ in the current direction, that is, change the state from $(x, y, \alpha)$ to the new state $(x + d\cos\alpha, y + d\sin\alpha, \alpha)$, and draw a line from the current position to the new position (see Fig. 3(b)).
- "f": Same as F, but just move without drawing a line (see Fig. 3(c)).
- "+": Increase the turn angle by $\delta$, that is, change the state from $(x, y, \alpha)$ to $(x, y, \alpha + \delta)$ (see Fig. 3(c)).

- "$-$": Decrease the turn angle by $\delta$, that is, change the state from $(x, y, \alpha)$ to $(x, y, \alpha - \delta)$ (see Fig. 3(d)).

As an example, let us consider how to design a turtle-geometry L-system that generates the Koch Island from the material on fractals (see Fig. 4). Let $d = 1$ and $\delta = 90°$. Let us assume that we start in the lower-left corner of the square ($(x, y) = (0, 0)$) and the heading is to the east ($\alpha = 0$).

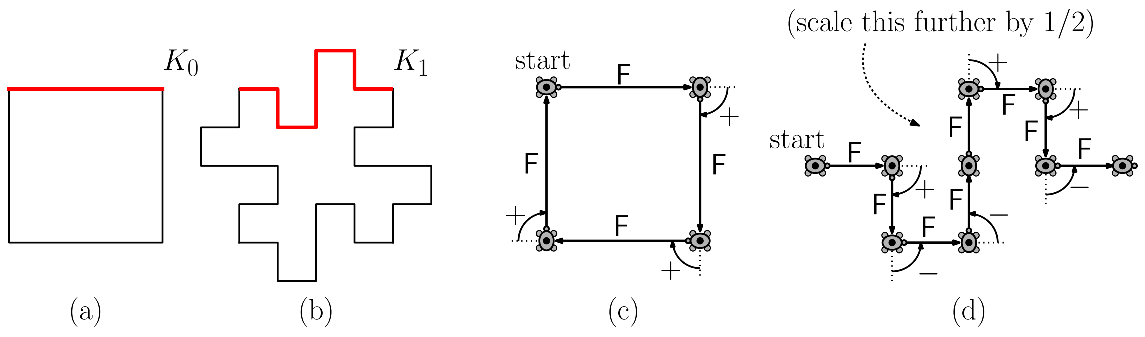


Fig. 4: The Koch Island.

$$\begin{aligned} V &= \{\mathsf{F}, \mathsf{f}, +, -\} \\ \omega &= \mathsf{F} + \mathsf{F} + \mathsf{F} + \mathsf{F} \\ p &= \{\mathsf{F} \rightarrow \mathsf{F} + \mathsf{F} - \mathsf{F} - \mathsf{FF} + \mathsf{F} + \mathsf{F} - \mathsf{F}\} \end{aligned}$$

Observe that $\omega$ generates a unit square (four line segments with 90° turns between each, see Fig. 3(b)). Then with each subsequent level we replace each existing line segment (F) with 8 segments, with appropriate turns in between (see Fig. 3(c)). To match the scale of earlier example, we can adjust $d_i = 1/4^i$ in order to generate $K_i$. This reflects the fact that with each iteration, the lengths of the line segments decreases by a factor of 1/4.

**Turtle-Based Trees:** Let us extend this to the generation of tree-like objects. To make this possible, we will introduce two special symbols "[" and "]" which intuitively mean respectively to save the current state on a push-down stack and to pop the stack and restore this as the current state. Such a system is called an *L-system with brackets.*

Let's see if we can apply this for generating a turtle-geometry drawing of a simple tree-like structure shown in Fig. 5. Intuitively, we identify the symbol "0" with a small stem and a circular leaf. This will be our starting tree, that is $\omega = 0$.

To "grow" the tree, we will generate a stem, which will be modeled by a line segment of unit length in the current direction, and will be denoted in our system by the symbol "1". The first-level tree consists of a stem with two copies of the leaf unit, each of half the original size, one rotated by 45° and the other rotated by $-45°$. To obtain this, we define two state-control symbols, "+", which scales by roughly 1/2 and rotates CCW by $-45°$ and "$-$", which scales by roughly 1/2 and rotates CW by 45°. Thus, our first-level tree can be described by the

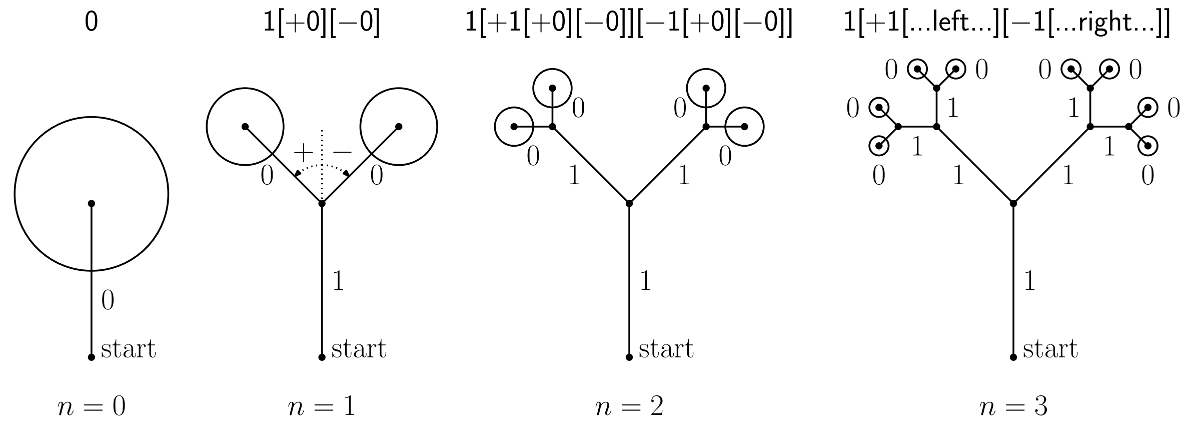


Fig. 5: A simple tree-like structure generated by an L-system with brackets. (Note that the figure is not drawn exactly to scale.) With each branching, the scale factor decreases by roughly 1/2.

string "1 [ + 0] [ − 0] ." The next level arises by replacing each of the leaf structures in the same recursive manner. This suggests the following L-system:

$$\begin{aligned} V &= \{0, 1, +, -, [, ]\} \\ \omega &= 0 \\ P &= \{0 \to 1[+0][-0]\} \end{aligned}$$

If we carry out the first few stages of the expansion, we have the following sequences. The associated sequence of drawings is shown in Fig. 5.

$$\begin{aligned} n=0 &: \ 0 \\ n=1 &: \ 1[+0][-0] \\ n=2 &: \ 1[+1[+0][-0]][-1[+0][-0]] \\ n=3 &: \ 1[+1[+1[+0][-0]][-1[+0][-0]]][-1[+1[+0][-0]][-1[+0][-0]]] \end{aligned}$$

**Randomization and Stochastic L-Systems:** As described, L-systems generate objects that are too regular to model natural objects. However, it is an easy matter to add randomization.

The first way of introducing randomness is to randomize the graphical/geometric operations. For example, rather than mapping terminal symbols into fixed actions (e.g., draw a unit-length line segment), we could add some variation (e.g., draw a line segment whose length is a random value between 0.9 and 1.1). Examples include variations in drawing lengths, variations in branching angles, and variations in thickness and/or texture (see Fig. 1).

While the above modifications alter the geometric properties of the generated objects, the underlying structure is still the same. We can modify L-systems to generate random structures by associating each production rule with a probability, and apply the rules randomly according to these probabilities. For example consider the following two rules:

$$\mathsf{a} \longrightarrow_{[0.4]} \mathsf{a}[\mathsf{b}] \qquad \mathsf{a} \longrightarrow_{[0.6]} \mathsf{b}[\mathsf{a}]\mathsf{b}$$

The interpretation is that the first rule is to by applied 40% of the time and the second rule 60% of the time.