L-system

L-system trees form convincing models of natural patterns

An **L-system** or **Lindenmayer system** is a **parallel rewriting system**, namely a variant of a formal grammar, most famously used to model the growth processes of plant development, but also able to model the morphology of a variety of organisms.[1] An L-system consists of an alphabet of symbols that can be used to make strings, a collection of production rules which expand each symbol into some larger string of symbols, an initial "axiom" string from which to begin construction, and a mechanism for translating the generated strings into geometric structures. L-systems can also be used to generate self-similar fractals such as iterated function systems. L-systems were introduced and developed in 1968 by the Hungarian theoretical biologist and botanist from the University of Utrecht, Aristid Lindenmayer (1925–1989).

Origins

'Weeds', generated using an L-system in 3D.

As a biologist, Lindenmayer worked with yeast and filamentous fungi and studied the growth patterns of various types of algae, such as the blue/green bacteria *Anabaena catenula*. Originally the L-systems were devised to provide a formal description of the development of such simple multicellular organisms, and to illustrate the neighbourhood relationships between plant cells. Later on, this system was extended to describe higher plants and complex branching structures.

L-system structure

The recursive nature of the L-system rules leads to self-similarity and thereby fractal-like forms are easy to describe with an L-system. Plant models and natural-looking organic forms are easy to define, as by increasing the recursion level the form slowly 'grows' and becomes more complex. Lindenmayer systems are also popular in the generation of artificial life.

L-system grammars are very similar to the semi-Thue grammar (see Chomsky hierarchy). L-systems are now commonly known as *parametric* L systems, defined as a tuple

$$\mathbf{G} = (V, \omega, P),$$

where

- **V** (the *alphabet*) is a set of symbols containing elements that can be replaced (*variables*)
- **ω** (*start*, *axiom* or *initiator*) is a string of symbols from **V** defining the initial state of the system
- **P** is a set of *production rules* or *productions* defining the way variables can be replaced with combinations of constants and other variables. A production consists of two strings, the *predecessor* and the *successor*. For any symbol A in V which does not appear on the left hand side of a production in P, the identity production $A \rightarrow A$ is assumed; these symbols are called *constants* or *terminals*.

The rules of the L-system grammar are applied iteratively starting from the initial state. As many rules as possible are applied simultaneously, per iteration; this is the distinguishing feature between an L-system and the formal language generated by a formal grammar. If the production rules were to be applied only one at a time, one would quite simply generate a language, rather than an L-system. Thus, L-systems are strict subsets of languages.

An L-system is *context-free* if each production rule refers only to an individual symbol and not to its neighbours. Context-free L-systems are thus specified by either a prefix grammar, or a regular grammar. If a rule depends not only on a single symbol but also on its neighbours, it is termed a *context-sensitive* L-system.

If there is exactly one production for each symbol, then the L-system is said to be *deterministic* (a deterministic context-free L-system is popularly called a *D0L-system*). If there are several, and each is chosen with a certain probability during each iteration, then it is a *stochastic* L-system.

Using L-systems for generating graphical images requires that the symbols in the model refer to elements of a drawing on the computer screen. For example, the program *Fractint* uses turtle graphics (similar to those in the Logo programming language) to produce screen images. It interprets each constant in an L-system model as a turtle command.

Examples of L-systems

Example 1: Algae

Lindenmayer's original L-system for modelling the growth of algae.

variables : A B

constants : none

start : A

rules : $(A \rightarrow AB)$, $(B \rightarrow A)$

which produces:

$n = 0$: A

$n = 1$: AB

$n = 2$: ABA

$n = 3$: ABAAB

n = 4 : ABAABABA

n = 5 : ABAABABAABAAB

n = 6 : ABAABABAABAABABAABABA

n = 7 : ABAABABAABAABABAABABAABAABABAABAAB

Example 1: Algae, explained

```
n=0:          A           start (axiom/initiator)
             / \
n=1:        A   B         the initial single A spawned into AB by rule (A → AB), rule (B → A) couldn't be applied
           /|    \
n=2:      A B     A       former string AB with all rules applied, A spawned into AB again, former B turned into A
         /| |     |\
n=3:    A B A     A B     note all A's producing a copy of themselves in the first place, then a B, which turns ...
       /| | |\    |\ \
n=4: A B A A B   A B A   ... into an A one generation later, starting to spawn/repeat/recurse then
```

If we count the length of each string, we obtain the famous Fibonacci sequence of numbers (skipping the first 1, due to our choice of axiom):

1 2 3 5 8 13 21 34 55 89 ...

For each string, if we count the *k*-th position from the left end of the string, the value is determined by whether a multiple of the golden mean falls within the interval $(k-1, k)$. The ratio of A to B likewise converges to the golden mean.

This example yields the same result (in terms of the length of each string, not the sequence of As and Bs) if the rule *(A → AB)* is replaced with *(A → BA)*, except that the strings are mirrored.

This sequence is a locally catenative sequence because $G(n) = G(n-1)G(n-2)$, where $G(n)$ is the *n*-th generation.

Example 2

- **variables** : 0, 1
- **constants**: [,]
- **axiom** : 0
- **rules** : (1 → 11), (0 → 1[0]0)

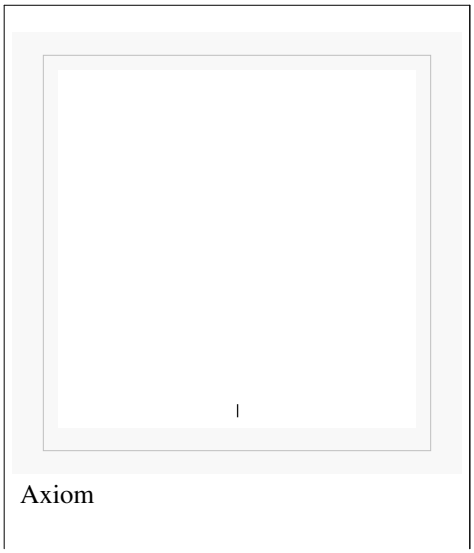
The shape is built by recursively feeding the axiom through the production rules. Each character of the input string is checked against the rule list to determine which character or string to replace it with in the output string. In this example, a '1' in the input string becomes '11' in the output string, while '[' remains the same. Applying this to the axiom of '0', we get:

axiom:	0
1st recursion:	1[0]0
2nd recursion:	11[1[0]0]1[0]0
3rd recursion:	1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0
…	

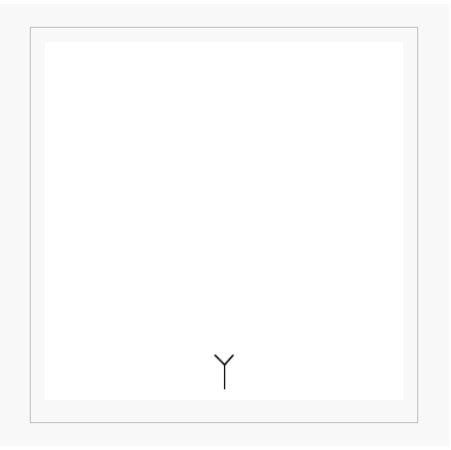
We can see that this string quickly grows in size and complexity. This string can be drawn as an image by using turtle graphics, where each symbol is assigned a graphical operation for the turtle to perform. For example, in the sample above, the turtle may be given the following instructions:

- 0: draw a line segment ending in a leaf
- 1: draw a line segment
- [: push position and angle, turn left 45 degrees
-]: pop position and angle, turn right 45 degrees

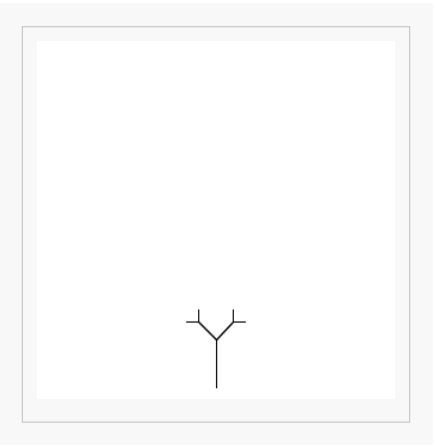
The push and pop refer to a LIFO stack (more technical grammar would have separate symbols for "push position" and "turn left"). When the turtle interpretation encounters a '[', the current position and angle are saved, and are then restored when the interpretation encounters a ']'. If multiple values have been "pushed," then a "pop" restores the most recently saved values. Applying the graphical rules listed above to the earlier recursion, we get:



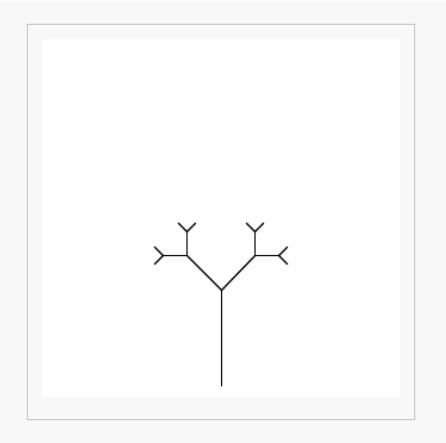
Axiom



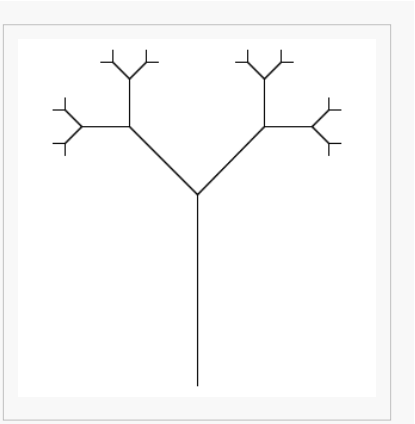
First recursion



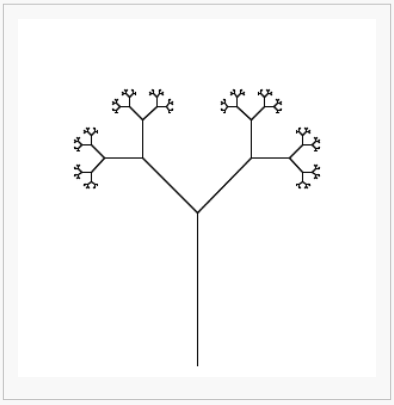
Second recursion



Third recursion



Fourth recursion



Seventh recursion, scaled down ten times

Example 3: Cantor dust

variables : A B

constants : none

start : A {starting character string}

rules : (A → ABA), (B → BBB)

Let *A* mean "draw forward" and *B* mean "move forward".

This produces the famous Cantor's fractal set on a real straight line **R**.

Example 4: Koch curve

A variant of the Koch curve which uses only right angles.

variables : F

constants : + −

start : F

rules : (F → F+F−F−F+F)

Here, F means "draw forward", + means "turn left 90°", and − means "turn right 90°" (see turtle graphics).

n = 0:

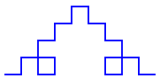
F

n = 1:

F+F−F−F+F

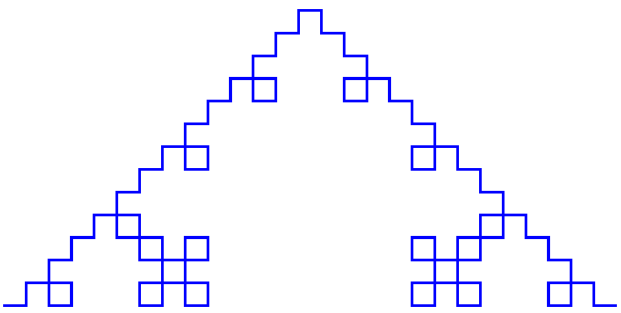
n = 2:

F+F−F−F+F + F+F−F−F+F − F+F−F−F+F − F+F−F−F+F + F+F−F−F+F



n = 3:

F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F +
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F −
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F −
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F +
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F



Example 5: Sierpinski triangle

The Sierpinski triangle drawn using an L-system.

variables : A B

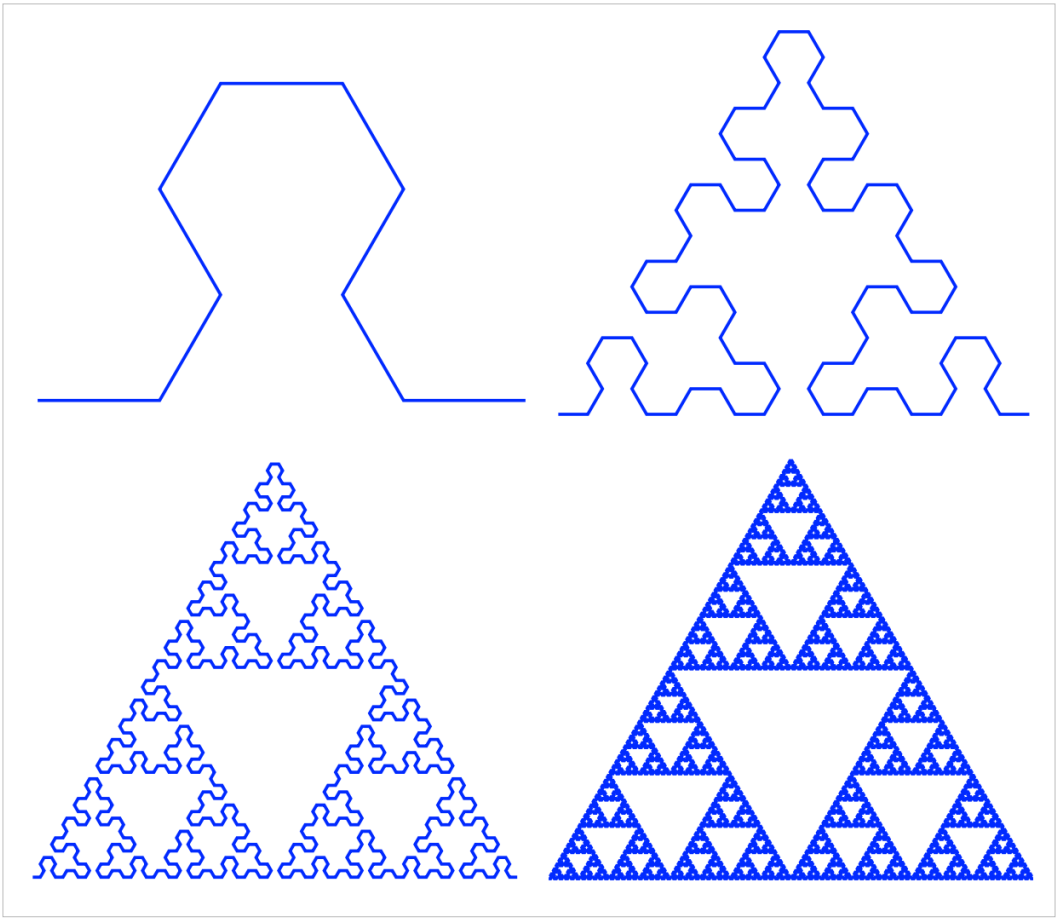
constants : + −

start : A

rules : (A → B−A−B), (B → A+B+A)

angle : 60°

Here, A and B both mean "draw forward", + means "turn left by angle", and − means "turn right by angle" (see turtle graphics). The angle changes sign at each iteration so that the base of the triangular shapes are always in the bottom (otherwise the bases would alternate between top and bottom).



Evolution for $n = 2$, $n = 4$, $n = 6$, $n = 8$ There is another way to draw the Sierpinski triangle using an L-system.

variables : F G

constants : + −

start : F−G−G

rules : (F → F−G+F+G−F), (G → GG)

angle : 120°

Here, F and G both mean "draw forward", + means "turn left by angle", and − means "turn right by angle".

Example 6: Dragon curve

The dragon curve drawn using an L-system.

variables : X Y

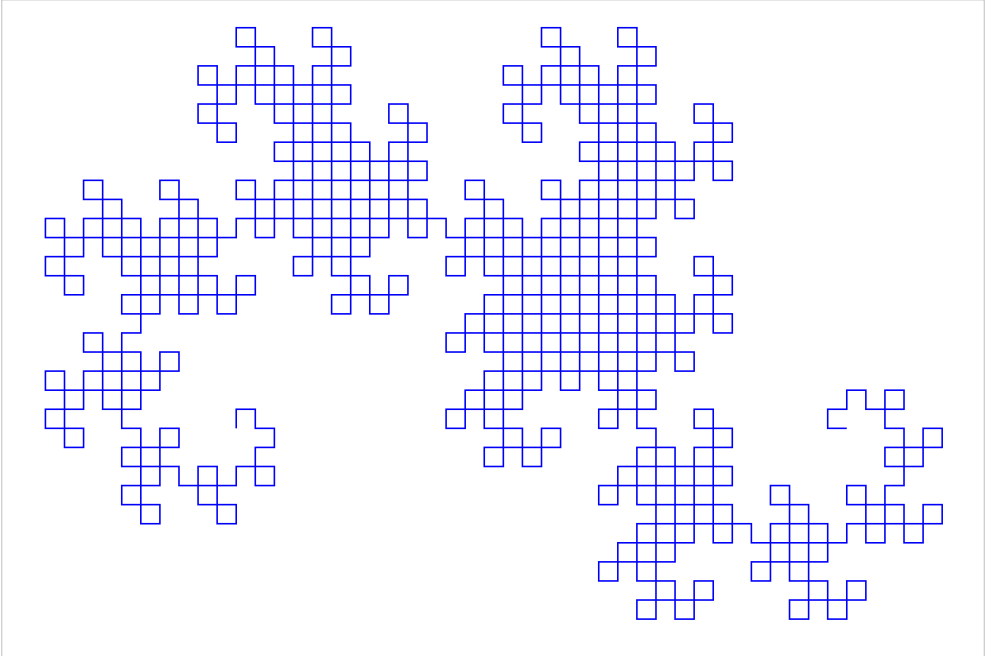
constants : F + −

start : FX

rules : (X → X+YF), (Y → FX-Y)

angle : 90°

Here, F means "draw forward", - means "turn left 90°", and + means "turn right 90°". X and Y do not correspond to any drawing action and are only used to control the evolution of the curve.



Dragon curve for *n* = 10

Example 7: Fractal plant

variables : X F

constants : + − [ ]

start : X

rules : (X → F-[[X]+X]+F[+FX]-X), (F → FF)

angle : 25°

Here, F means "draw forward", - means "turn left 25°", and + means "turn right 25°". X does not correspond to any drawing action and is used to control the evolution of the curve. **[** corresponds to saving the current values for position and angle, which are restored when the corresponding **]** is executed.

Fractal plant for $n = 6$

Variations

A number of elaborations on this basic L-system technique have been developed which can be used in conjunction with each other. Among these are stochastic, context sensitive, and parametric grammars.

Stochastic grammars

The grammar model we have discussed thus far has been deterministic—that is, given any symbol in the grammar's alphabet, there has been exactly one production rule, which is always chosen, and always performs the same conversion. One alternative is to specify more than one production rule for a symbol, giving each a probability of occurring. For example, in the grammar of Example 2, we could change the rule for rewriting "0" from:

$0 \rightarrow 1[0]0$

to a probabilistic rule:

$0\ (0.5) \rightarrow 1[0]0$

$0\ (0.5) \rightarrow 0$

Under this production, whenever a "0" is encountered during string rewriting, there would be a 50% chance it would behave as previously described, and a 50% chance it would not change during production. When a stochastic grammar is used in an evolutionary context, it is advisable to incorporate a random seed into the genotype, so that the stochastic properties of the image remain constant between generations.

Context sensitive grammars

A context sensitive production rule looks not only at the symbol it is modifying, but the symbols on the string appearing before and after it. For instance, the production rule:

$b < a > c \rightarrow aa$

transforms "a" to "aa", but only If the "a" occurs between a "b" and a "c" in the input string:

…bac…

As with stochastic productions, there are multiple productions to handle symbols in different contexts. If no production rule can be found for a given context, the identity production is assumed, and the symbol does not change on transformation. If context-sensitive and context-free productions both exist within the same grammar, the context-sensitive production is assumed to take precedence when it is applicable.

Parametric grammars

In a parametric grammar, each symbol in the alphabet has a parameter list associated with it. A symbol coupled with its parameter list is called a module, and a string in a parametric grammar is a series of modules. An example string might be:

a(0,1)[b(0,0)]a(1,2)

The parameters can be used by the drawing functions, and also by the production rules. The production rules can use the parameters in two ways: first, in a conditional statement determining whether the rule will apply, and second, the production rule can modify the actual parameters. For example, look at:

$a(x,y) : x = 0 \rightarrow a(1, y+1)b(2,3)$

The module a(x,y) undergoes transformation under this production rule if the conditional x=0 is met. For example, a(0,2) would undergo transformation, and a(1,2) would not.

In the transformation portion of the production rule, the parameters as well as entire modules can be affected. In the above example, the module b(x,y) is added to the string, with initial parameters (2,3). Also, the parameters of the already existing module are transformed. Under the above production rule,

a(0,2)

Becomes

a(1,3)b(2,3)

as the "x" parameter of a(x,y) is explicitly transformed to a "1" and the "y" parameter of a is incremented by one.

Parametric grammars allow line lengths and branching angles to be determined by the grammar, rather than the turtle interpretation methods. Also, if age is given as a parameter for a module, rules can change depending on the age of a plant segment, allowing animations of the entire life-cycle of the tree to be created.

Open problems

There are many open problems involving studies of L-systems. For example:

- Characterisation of all the deterministic context-free L-systems which are locally catenative. (A complete solution is known only in the case where there are only two variables).
- Given a structure, find an L-system that can produce that structure.

Types of L-systems

L-systems on the real line **R**:

- Prouhet-Thue-Morse system

Well-known L-systems on a plane $\mathbf{R}^2$ are:

- space-filling curves (Hilbert curve, Peano's curves, Dekking's church, kolams),
- median space-filling curves (Lévy C curve, Harter-Heighway dragon curve, Davis-Knuth terdragon),
- tilings (sphinx tiling, Penrose tiling),
- trees, plants, and the like.

Books

- Przemyslaw Prusinkiewicz, Aristid Lindenmayer - The Algorithmic Beauty of Plants PDF version available here for free [2]
- Grzegorz Rozenberg, Arto Salomaa - *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology* ISBN 978-3-540-55320-5
- D.S. Ebert, F.K. Musgrave, et al. - *Texturing and Modeling: A Procedural Approach*, ISBN 0-12-228730-4

Notes

[1] Grzegorz Rozenberg and Arto Salomaa. The mathematical theory of L systems (Academic Press, New York, 1980). ISBN 0-12-597140-0
[2] http://algorithmicbotany.org/papers/#abop

External links

- Algorithmic Botany at the University of Calgary (http://algorithmicbotany.org/)
- Branching: L-system Tree (http://www.mizuno.org/applet/branching/) A Java applet and its source code (open source) of the botanical tree growth simulation using the L-system.
- Fractint L-System True Fractals (http://spanky.triumf.ca/www/fractint/lsys/truefractal.html)
- "powerPlant" an open-source landscape modelling software (http://sourceforge.net/projects/pplant/)
- *Fractint* home page (http://www.fractint.org/)
- A simple L-systems generator (Windows) (http://www.generation5.org/content/2002/lse.asp)
- Lyndyhop: another simple L-systems generator (Windows & Mac) (http://lyndyhop.sourceforge.net/)
- An evolutionary L-systems generator (anyos*) (http://www.cs.ucl.ac.uk/staff/W.Langdon/pfeiffer.html)
- "LsystemComposition" (http://www.pawfal.org/index.php?page=LsystemComposition). Page at Pawfal ("poor artists working for a living") about using L-systems and genetic algorithms to generate music.
- eXtended L-Systems (XL), Relational Growth Grammars, and open-source software platform GroIMP. (http://www.grogra.de/)
- A JAVA applet with many fractal figures generated by L-systems. (http://to-campos.planetaclix.pt/fractal/plantae.htm)
- Musical L-systems: Theory and applications about using L-systems to generate musical structures, from waveforms to macro-forms. (http://www.modularbrains.net/support/SteliosManousakis-Musical_L-systems.

pdf)

- Online experiments with L-Systems using JSXGraph (JavaScript) (http://jsxgraph.uni-bayreuth.de/wiki/index.php/L-systems)
- Flea (http://flea.sourceforge.net/) A Ruby implementation of LSYSTEM, using a Domain Specific Language instead of terse generator commands
- Lindenmayer power (http://madflame991.blogspot.com/p/lindenmayer-power.html) A plant and fractal generator using L-systems (JavaScript)
- Rozenberg, G.; Salomaa, A. (2001), "L-systems" (http://www.encyclopediaofmath.org/index.php?title=L-systems&oldid=14908), in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Laurens Lapré's L-Parser (http://laurenslapre.nl/lapre_004.htm)
- HTML5 L-Systems - try out experiments online (http://www.kevs3d.co.uk/dev/lsystems/)
- The vector-graphics program (http://inkscape.org/) Inkscape features an L-System Parser

Article Sources and Contributors

L-system *Source*: http://en.wikipedia.org/w/index.php?oldid=528548898 *Contributors*: Altales Teriadem, Arichnad, Ascánder, Atoll, B.d.mills, Babajobu, Ben Standeven, Bhickey, Bryan Derksen, Burn, CBM, Canderson7, Chiswick Chap, Chopchopwhitey, Citronglass, Cjthellama, Cncplayer, Composingliger, DFRussia, Deerwood, Dmmd123, Długosz, Extro, Fremerl, Gabriele ricci, Gandalf61, Giftlite, Helohe, Hirzel, Hyacinth, IanOsgood, ItzSteeven, Ivan Štambuk, Ivokabel, JYOuyang, Jason Davies, Jdandr2, Jeronimo, JoelCastellanos, Jpbowen, Kainino, Kate, Kevyn, Kku, Kotasik, LC, Linas, Lomacar, Lowellian, Lukax, LutzL, Marektad, Michael Hardy, Mike Serfas, Mikhail Ryazanov, Mizuno, MokoMull, Mukkakukaku, Nevit, Oderbolz, Oleg Alexandrov, P0lyglut, PaulTanenbaum, Piano non troppo, Plrk, Pmcculler, Prophile, RDBury, RPHv, Rdsmith4, Robertinventor, Sakurambo, SchuminWeb, Shadow600, Solkoll, Spencerk, StefanVanDerWalt, Sun Creator, Svea Kollavainen, Svick, ThomasPusch, Thunderbolt16, TimTim, Timwi, Tmcw, Tobias Bergemann, Tomasz J Kotarba, Treelo, Tó campos, Wassermann7, XJamRastafire, 119 anonymous edits

Image Sources, Licenses and Contributors

File:Dragon trees.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Dragon_trees.jpg *License*: unknown *Contributors*: Nevit, Solkoll

Image:Fractal weeds.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Fractal_weeds.jpg *License*: unknown *Contributors*: Nevit, Solkoll, 1 anonymous edits

Image:Graftal0.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal0.png *License*: Creative Commons Attribution 3.0 *Contributors*: Svick

Image:Graftal1.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal1.png *License*: Creative Commons Attribution 3.0 *Contributors*: Svick

Image:Graftal2.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal2.png *License*: Creative Commons Attribution 3.0 *Contributors*: Svick

Image:Graftal3.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal3.png *License*: Creative Commons Attribution 3.0 *Contributors*: Svick

Image:Graftal4.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal4.png *License*: Creative Commons Attribution 3.0 *Contributors*: Svick

Image:Graftal7.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Graftal7.png *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: Svick

Image:Cantor set in seven iterations.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Cantor_set_in_seven_iterations.svg *License*: Public Domain *Contributors*: This image is valid SVG

image:square koch.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Square_koch.svg *License*: Public Domain *Contributors*: Composingliger

image:square koch 1.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Square_koch_1.svg *License*: Public Domain *Contributors*: Composingliger

image:square koch 2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Square_koch_2.svg *License*: Public Domain *Contributors*: Composingliger

image:square koch 3.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Square_koch_3.svg *License*: Public Domain *Contributors*: Composingliger

Image:Serpinski Lsystem.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Serpinski_Lsystem.svg *License*: Public Domain *Contributors*: User:Arichnad (svg)António Miguel de Campos (jpg original)

Image:Dragon curve L-system.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Dragon_curve_L-system.svg *License*: Public Domain *Contributors*: António Miguel de Campos

Image:Fractal-plant.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Fractal-plant.svg *License*: Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors*: Sakurambo

License