L-Systems Construction Kit

BENJAMIN D. FARKAS BENFARKAS@ALUM.RPI.EDU
NICK ROMANYSHYN NICHOLAS.ROMANYSHYN@COLORADO.EDU
PATRICK CLARY CLARY@COLORADO.EDU

DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF COLORADO, BOULDER, CO 80309, USA

Abstract

1. Introduction

L-Systems were invented by Aristid Lindenmayer in the 1960's [Lindenmayer, 1968]. Lindemeyer noticed that complex biological plants and structures had recursive patterns and could be compactly represented through simple grammars (strings of text), called L-Systems. Lindemeyer published a book, *The Algorithmic Beauty of Plants* [Prusinkiewica and Lindenmayer, 1990], where he displays the raw power of these simple L-Systems by producing beautifully magnificent plants and structures. Some examples of these beautiful structures from Lindemeyers original text are displayed in Figure 1.1.

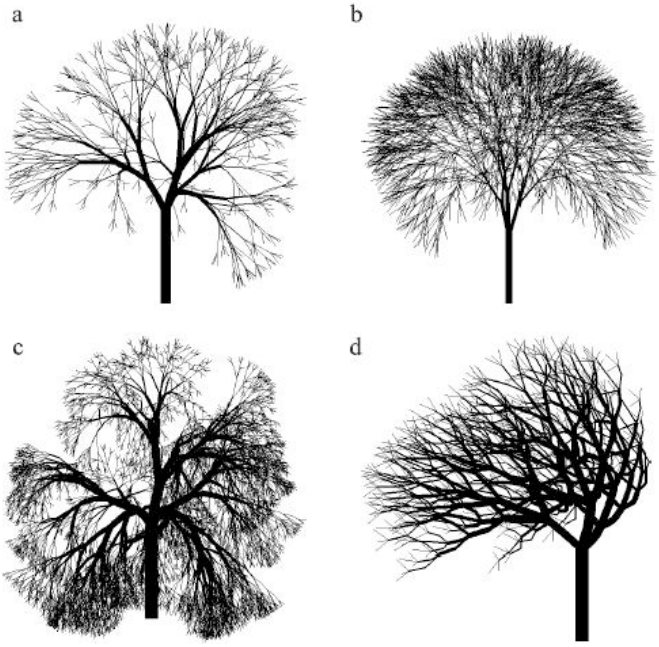

Figure 1.1 – Examples of L-Systems structures produced by Lindenmeyer and published in his famous book *The Algorithmic Beauty of Plants* [Prusinkiewica and Lindenmayer, 1990].

Traditionally, L-Systems have been used to produce very complex graphical structures on a computing device and displayed on the computer screen. We seek to produce these structures as physical objects for people to learn from and experiment with. Our approach involves several components including: L-systems simulation software, device for converting the L-system to HPGL (readable by Corel Draw), and a laser-printed output of the L-System with constructible pieces, where the L-System allows. Thus, we seek to produce a construction kit version of L-Systems, allowing someone to play around and model the structure using computer software and print it to a laser-cutter (or 3-D printer). This can be used to create three types of objects: an etched drawing, a stencil, or individual construction pieces. As you will see later, each of these is useful for different types of L-system. Thus, the user has the option of which one of these he/she wishes to produce.

2. L-System Construction Kit Overview

The alphabet for the L-Systems in which we implemented contain 20 characters, dealing with drawing lines, polygons, pushing information onto a stack, poping information off of a stack, manipulating color and line thickness. An overview of our L-System alphabet is given in Appendix A.

An L-System input file is produced that contains the grammar and relevant input parameters. The format of this file is detailed in Appendix B.

The input file is supplied to the L-System Simulator software which interprets the grammar and draws the structure to the computer screen. Using this tool, a user may preview the structure prior to printing it to the laser printer (or 3D printer). Examples of output generated by the L-System simulator are given in Figure 2.1.

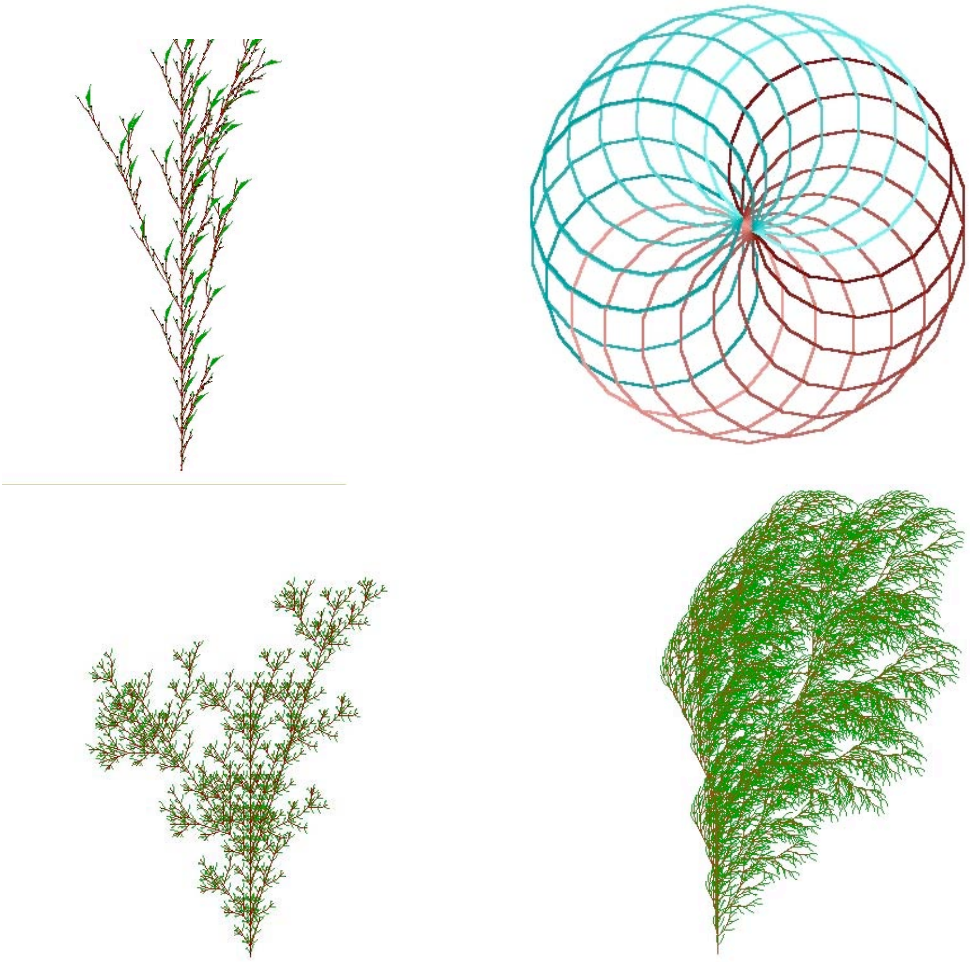

Figure 2.1 – Examples of L-Systems structures produced by the L-System Simulator described in this paper.

As the L-System is being generated using the L-System simulator and displayed on screen, an HPGL file is being produced which can be interpreted by *Corel Draw* and then printed to a laser cutter. Information about the HPGL file format may be found at: http://lprng.sourceforge.net/DISTRIB/RESOURCES/HPGL_short_summary.html. Using this HPGL file, *Corel Draw* is capable of displaying and communicating with the *Versa Laser* laser-cutter. While many types of media are applicable, wood and thick paper were used for the L-systems described in this paper.

3. Implementation Details

In order to implement this L-system simulator, each letter in the alphabet must be implemented. Based on the number of iterations (input in the input-file), a string of some number of commands (letters) is produced. The implementation of an interesting (but small) subset of the alphabet is described here.

F – draw a line of length <line length> and of a specified number of degrees (determined by the L-System.

With the F command, we wish to draw a line of a specified length and a specified number of degrees onto the screen. This is translated to HPGL by geometrically calculating the projected (x,y) location of the end of the line segment. The PD (pen down) command is used to draw a line from the current location to a specified (x,y) location. Thus, PD end_x, end_y; is used here.

@: draw a circle with diameter equal to the current line length

The @ command is used to draw a circle on the screen at the current (x,y) locations with a specified radius. This command can be translated into HPGL by the CI radius command.

{: save current turtle to be used to draw a filled polygon at a later time

A turtle contains relevant information about the current location, line thickness, and color. This information can be saved and some number of these [saved] turtles can be used to produce a complex polygon.

}: draw the filled polygon, based on all the saved turtles

Once the appropriate amount of turtles has been saved, a complex polygon can be produced. Naturally, the polygon is produced (and filled) by drawing the sequence of lines (saved by the turtles) using the F command.

These commands represent some of the major commands of the L-system. Using these commands very complex structures can be produced. L-systems also contain color directives for manipulating the color of the structure. This can be used to produce beautifully complex and realistic objects. While there is no color per se in the laser cutter world, we had many ideas for how to deal with this.

If the L-system is to be etched then we can use laser speed and power to simulate color. *Versa Laser* allows for eight pre-set colors. We utilized these colors and interpolated the RGB values of the program into one of these pre-sets. <MORE ON THIS HERE> This produces varying cuts and one can clearly see that there is an intended difference in the look of the cuts.

4. Implementation Difficulties

A major difficulty occurred in interfacing the laser cutter. We had a large HPGL file (~15K commands) produced by the L-system simulator and the laser cutter would always freeze at the same point approximately 15% into the cut. We could not figure out why this error occurred so we decided upon the non-elegant solution of breaking the HPGL file into several [smaller] files. This created a couple problems since we now could not manually scale our L-system drawings using *Corel Draw*. Therefore, the files needed to be scaled by a certain factor because 1000x1000 represents one square inch in *Corel Draw*. However, once these problems were overcome the laser cutter did not freeze with the multiple-file solution!

Another difficulty was in converting the drawing language (TOOGL: CMUGraphics library [http://www.cs.duke.edu/csed/tapestry/graphics.html]) of the L-system simulator to HPGL. While some conversions were straight forward, others proved difficult since we needed to translate to their coordinate system and calculate ending (x,y) locations (where you can pass degrees to the drawing utility). It took a couple of days to get all of this right but once we did it worked well.

Converting the line-based L-system into 2D block-based construction kit components was non-trivial. Each line must be converted into a block of a specified width or else the laser cutter will produce an etching or a stencil (cuts through the wood). Also, figuring out where [on the L-system] to produce snap-able pieces was a problem. We produced a reasonable block-conversion-system and allow the user to manually specify where he/she would like to produce pieces.

5. Results

As mentioned earlier, there are possibly three different types of objects that can be produced. First, an etching of the L-system can be produced on any type of medium (within reason of course). Second, a stencil of the L-system can be made by cutting all the way through the medium. Third, individual pieces can be produced that can later be constructed manually to produce the resulting L-system. One can use pieces from any L-system with pieces from any other L-system to produce hybrid pieces and new ideas.

5.1 Etching

Bush in a Hurricane

Figure 5.1 – a) The Bush in a Hurricane, produced by the L-system in Appendix C. b) The etching of the Bush in a Hurricane produced by the *Versa Laser* laser-cutter.

Multi-Color Spiral Kaleidoscope

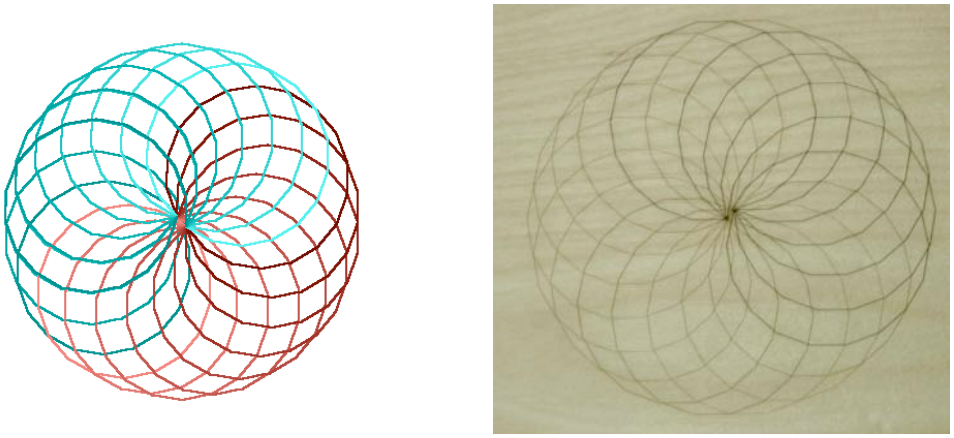

Figure 5.2 – a) The Multi-Color Spiral Kaleidoscope, produced by the L-system in Appendix C. b) The etching of the Multi-Color Spiral Kaleidoscope produced by the *Versa Laser* laser-cutter.

Spring Plant

Figure 5.3 – a) The Spring Plant, produced by the L-system in Appendix C. b) The etching of the Spring Plant produced by the *Versa Laser* laser-cutter.

5.2 Stenciling

When cut entirely through, the medium acts as a stencil for the L-system. There are two ways to accomplish this. The piece can be cut out and then trace on both sides creating a block-representation of the L-system using the innards of the piece. Another way is to trace the original wood after the piece has been cut out. Examples are given in Figure 5.4.

Figure 5.4 – The original wood piece with the cut-out stencil (left). A drawing using the stencil on the left (middle). The cut-out (block) stencil (right).

5.3 Construction Kit Pieces

To make modular connection pieces, we decided on the medium of wood. Using wood of a pre-set thickness allows us to create modular pieces that *snap in* to the wood. This allows for 3D constructions, since snaps can be places along the x,y, or z axes. Using these *snappers*, L-systems can be mixed-and-matched creating hybrid pieces. Examples of different types of *snappers* are given in Figure 5.5. They vary in size, each supporting a different number of branches.

Figure 5.5 – Construction kit components (*snappers*) for connecting L-system components together.

Using the *snapper* pieces, we built a 3-D tree (based on an L-system grammar). The tree is displayed in Figure 5.6.

Figure 5.6 – Constructed 3d L-system tree. Constructed by snapping together pieces of the L-system.

6. Educational and other Applications

L-systems are wonderfully magnificent and fascinating structures. It is interesting to think of objects in terms of L-systems. How can one define such complicated structures with such compact grammars? Are biological structures so simple that they boil down to an L-system? These are compelling questions that someone begins to ask when they start playing around with L-systems. The answers, however, are not easy. L-systems are a relatively new and advanced concept in science. They over a drastically different view of natural objects (like plants) and how simple they really might be (if implemented using the right grammars).

One can also learn about powerful concepts in the world (and computer science), such as recursion, stack-operations, and combinatorics. L-systems are also a great way to express oneself creatively. Instead of painting with a canvas a painter can now paint with a grammar. Make a small tweak to the grammar, preview and iteratively change. This is not really possible with traditional canvas-painting since there is no way to go back and iteratively change your piece without starting over.

L-systems can also be used for image compression. The bush in a hurricane (Figure 5.1) is compactly represented by the following:

Iterations: 5

Input File:

```
90 20 1 4 320 460 0 150 0
F
F BFF-[->GF+>F+F]B+[+>GF-F-F]
;
```

The total representation of the L-system takes roughly 60 characters (60B), whereas an entire BMP image is about 1.4MB on disk. This is a 99.9996% reduction in size. It is not hard to imagine L-system simulators (like the one outlined in this paper) that can [quickly] interpret and display these complex images.

7. Conclusion

This paper presented the framework for an L-system simulator system which is capable of producing several types of object-representations from simple grammars. First, on-screen graphical representations of the objects are shown to the user. Naturally, it is best to tweak the L-system at this point. Then, once satisfied, the user is able to print an etched version of L-system on one of several types of media. The user is also able to print a stencil representation of the object. Lastly, key components of the object can be cut as individual building pieces and *snapped* together with other pieces to form new 3D L-systems.

This framework is a new type of construction kit system – one in which the construction can be simulated or previewed on a computer screen and then produced in the physical world. Users will become curious with how and why these L-systems work. This will lead to better understand of key concepts such as recursion and combinatorics. We believe that this system is a new concept, fun, challenging, and educational.

References

[Lindenmayer, A., 1968] *Mathematical models for cellular interaction in development*, Journal of Theoretical Biology, 18:280-315.

[Prusinkiewica, P., and Lindenmayer, A., 1990] The algorithmic beauty of plants. Springer-Verlag. ISBN: 0387972978

Appendix A – L-System Alphabet Description

- **F:** draw a line of length <line length> and of a specified number of degrees (determined by the L-System.

- **E:** same as E above. It is good to have two such draw symbols so that you can expand (have a production rule) for one and not the other.
- **f:** advance a number of pixels based on <line length> however do not draw the line.
- **@:** draw a circle with diameter equal to the current line length
- **+:** change degrees clockwise according to <degrees rotate>
- **-:** change degrees counter-clockwise according to <degrees rotate>
- **#:** increase thickness one pixel
- **!:** decrease thickness by one pixel, if thickness is at 0, it will stay at 0.
- **`:** increments the current color
- **‘:** decrements current color
- **R:** changes current color to red. R=150, G=0, B=0, medium red shade.
- **G:** changes current color to green. R=0, G=150, B=0, medium green shade.
- **B:** changes current color to brown. R=139;G=69;B=19, medium brown shade.
- **L:** changes current color to black. R=0; G=0; B=0.
- **[:** push current turtle onto the stack (turtle).
- **]:** pop current turtle from the stack and return to those values that were on the stack.
- **{:** save current turtle to be used to draw a filled polygon
- **}:** draw the filled polygon, based on all the saved turtles
- **>:** clockwise half the number of degrees as -
- **<:** counter-clockwise half the number of degrees as +

Appendix B – L-System Input File Details

The first line of the file must contain the following initial elements:
•<degrees to start> - an integer
•<degrees rotate> - integer, how many degrees to change
•<starting thickness> -integer, for most objects a thickness of 1 or 2 looks best, measured in pixels.
•<line length> - integer – amount to draw the line each time in pixels
•<starting x> - integer - initial position for x, screen coordinate, based on a 480x640 window <starting y> - integer - initial position for x, screen coordinate, based on a 480x640 window
•<initial R> - integer from 0-255, amount of red in color
•<initial G> - integer from 0-255, amount of green in color
•<initial B> - integer from 0-255, amount of blue in color
•<starting string> - the axiom, ω - the starting string

Next, the production rules are listed in such a fashion:

•<left hand side> - a single character that is in the designed alphabet (described in III).
•<right hand side> - any number of characters the left hand side is to "produce" each iteration through (ie F->F[F]F etc).

• Finally, input files must end with the character **;** on a single line by itself, this is the termination symbol.

Appendix C – Example L-System Input Files

Bush in a Hurricane

Iterations: 5

Input File:

```
90 20 1 4 320 460 0 150 0
F
F BFF-[->GF+>F+F]B+[+>GF-F-F]
;
```

Multi-Color Spiral Kaleidoscope

Iterations: 20
Input File:

```
90 20 2 30 320 200 0 150 150
S
S [-CS]F+F+F+F+F+F+F+F+F+F+F+F+F+F+F+F+F+F+FC
C ````````````````
;
```

Spring Plant

Iterations: 4

Input File:

```
90 40 1 3 320 460 0 150 0
F
F R!F[+GF>F<@!][+GF<F@!][-GFF<F@!]RF!F
;
```