# Lindenmayer Systems – Experimenting with Software String Rewriting as an Assist to the Study and Generation of Architectural Form

*Antonio Serrato-Combe*
*College of Architecture + Planning*
*The University of Utah, United States of America*
*http://128.110.143.59*

***Abstract.*** *In 1968 Aristid Lindenmayer proposed a series of mathematical constructs as a foundation for an axiomatic theory of form development. Since that time, Lindenmayer Systems or L-systems have evolved and found many practical applications in the computer visualization area. Generation of fractal imagery, realistic modeling and high quality visualization of organic forms and even music generation are now possible with the assistance of L-systems.*
*But, is it possible to use L-systems in architectural design? Why would anyone use L-systems in architectural design? How would one use them? What could one expect from their use?*
*In addition to providing answers to the above questions this paper presents:*
*1. Concepts behind L-systems*
*2. The need to transform L-Systems so they can have creative architectural application possibilities*
*3. Examples on the architectural use of L-Systems*
*4. Conclusions*

***Keywords.*** *Form generation, Lindenmayer, String rewriting, Visualization*

## Lindenmayer Systems Introduction

Lindenmayer systems are computer software string rewriting techniques developed by Astrid Lindenmayer in 1968. Originally the techniques were used to model and visualize the morphology of a variety of organisms. For example, in the following set of rules one can look at the growth of the very simple organism Chaetomorpha Linum.

```
A -> DB           Axiom
B -> C   Growth rule
C -> D
D -> E
E -> A
```

The following is the growth pattern generated by this set of rules. This growth pattern matches the arrangement of cells in the original alga.

```
Stage  0 :                  A
Stage  1 :        D                  B
Stage  2 :        E                  C
Stage  3 :        A                  D
Stage  4 :     D       B             E
Stage  5 :     E       C             A
Stage  6 :     A       D         D      B
Stage  7 :   D     B    E        E      C
Stage  8 :   E     C    A        A      D
Stage  9 :   A     D   D   B   D   B    E
Stage 10 :  D    B   E E   C   E    C   A
Stage 11 :  E   C   A  A   D   A   D  D   B
```

## Example 2

The following is a more complex set of rules to generate a monopodial skeleton:

```
11
45
15
#
c(12)FFAL
#
A=~(10)F[&'(.8)!BL]>(137)'!(.9)A
B=~(15)F[-'(.8)!(.9)$CL]'!(.9)C
C=~(20)F[+'(.8)!(.9)$BL]'!(.9)B
#
L=~c(8){+(30)f(30)-(120)f(30)-(120)f(30)}
```

The visualization of these rules using a Java Applet looks like this:

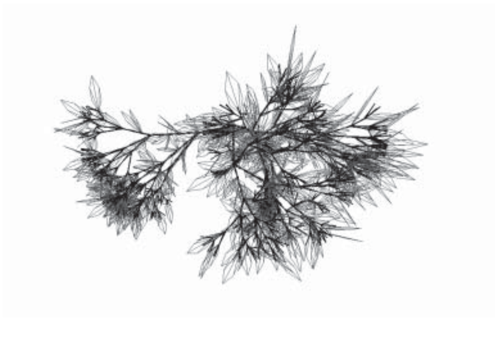


Figure 1. Monopodial Skeleton

## The Case for Applicability to Architecture

While the actual application of L-Systems to biological sciences, virtual ecosystems, population studies has been spectacular, efforts directed at architectural design have been minimal at best. This is because:

1. Some claim that a generative rule based, numerical method of mediation in architectural design is absurd and results in the worst kind of formalism.

2. Others find it difficult to understand and apply L-systems non-mathematical notation.

3. Not until recently it has been almost impossible to integrate architectural visualization to L-systems.

## Answers to the Issues Raised Above

1. L-systems, like other generative design approaches, can and should be used as assists to generate and visualize complex architectural form, not as sole design instruments.

2. With the assist of two new JAVA applets, it is now very easy to generate and visualize L-systems axioms.

3. The applets' output can be read by any CAD or NURB application resulting in stunning form creation and visualization.

## So, Why Should an Architect Use L-Systems in Her or His Work?

1. With L-systems architects can easily experiment and visualize a very unique and exciting array of forms and shapes that can be further manipulated with CAD.

2. L-systems can assist in visualizing how forms are derived and transformed.

3. L-systems can generate 2 and 3-D geometries in a matter of seconds that otherwise would take countless hours to generate using traditional

CAD approaches. Many of those forms would be practically impossible to generate without such an assist.

4. L-systems can be used to apply organic growth principles to architectural design.

5. A designer can begin with a very simple set of rules and then experiment with countless forms, variations and transformations, all with very quick and simple changes in the axiom writing.

6. They are fun to play with!

Experiments

The following are two experiments illustrating how the approach can assist designers in conceiving and visualizing potential architectural solutions.

## Experiment One

Let's assume that the designer wishes to generate a form that begins as a small unit and begins to grow as it rotates. This task is related to the design of an open space structure in front of a large hotel development. In this particular instance the designer is looking at 18 iterations. It is desired that every time a new iteration occurs the next iteration will rotate 15 degrees. These two numbers, 18 iterations and 15 degrees are just starting points. They can be modified at any time. The architectural concept behind these instructions is that of a series of spaces that will grow organically every time a new element is added to the composition. This very simple scheme would look like this:

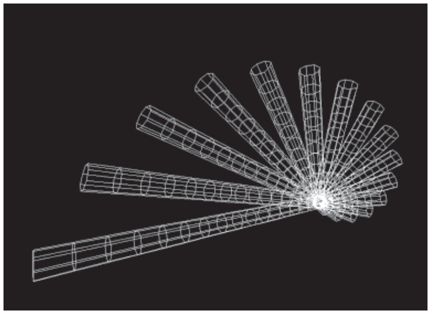


Figure 2. Experiment one.

The instruction file looks like this:

```
18            # recursion
15            # basic angle
100           # thickness
A             # axiom
A=B>(15)A     # rule 1 : add a branch and roll
10 deg right
B=[&(45)C]    # rule 2 : pitch down 45 deg and
start a branch
C=FC          # rule 3 : expand a branch with
a forward element
@             # end
```

Now, after looking at this scheme, the designer wishes to explore some other variation perhaps reducing the number of iterations to 12, increasing the angle to 20 degrees and making the individual elements a bit bigger. The actual operation would simply replace a few numbers. It would only take a few seconds. And, the result would look like this:

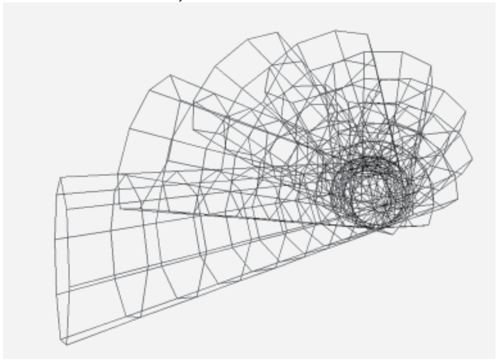


Figure 3. Variation on experiment one.

The LS file now looks like this:

```
12            # recursion
20            # basic angle
300           # thickness
A             # axiom
A=B>(20)A     # rule 1 : add a branch and roll
10 deg right
B=[&(45)C]    # rule 2 : pitch down 45 deg and
start a branch
C=FC          # rule 3 : expand a branch with
a forward element
@             # end
```

The designer could obviously spend additional time in generating more variations and refinements. Each new iteration would only take a few seconds. Assuming that he or she has arrived at an interesting approach that would have possibilities of addressing architectural program requirements, the designer would simply save this file and begin a process of transformation in any CAD application. In the next phase of the experimentation the edges of the preliminary investigation are given appropriate thickness in order to look at possible structural conformations.

And finally, the new scheme is inserted on the model of the development. The steps illustrated above and below only took a few minutes. No coordinate entry was required.

It is very important to note at this point that the design solution illustrated above is not the final solution. However, the approach has enabled the designer to look at design options, scales, size of elements, complexity of intervention and other design aspects without having spent considerable time and energies entering coordinates in order to generate the model. At this junction, the designer

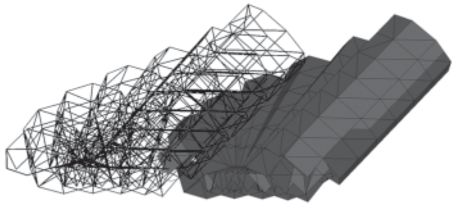


*Figure 4. Potential Architectural Conformation*

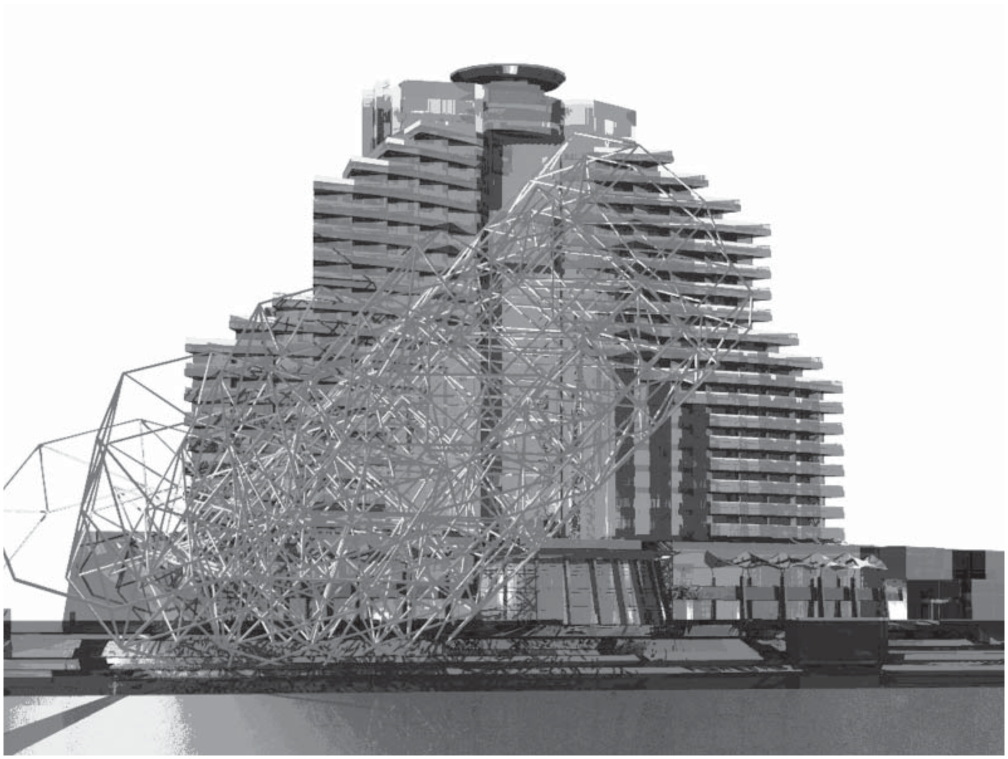


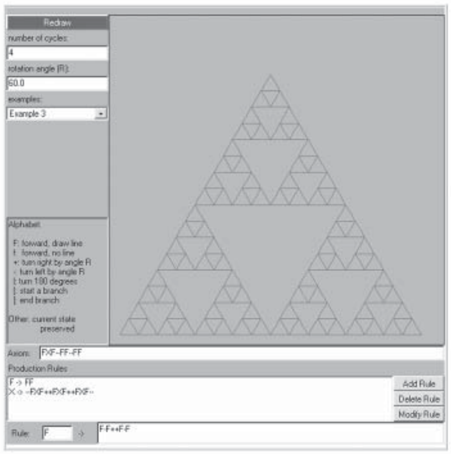
*Figure 5. Final result of experiment one.*

can certainly use the option of keeping the model as it stands, but can also generate another set of re-writing rules to test other alternatives. The benefits are obvious. From beginning to end, instead of spending considerable time and energies defining geometry every single element, time and energies are instead better spent analyzing the aesthetic and structural properties of the composition. In this way many possibilities are explored in order to arrive at the best solution.

## Experiment Two

In this imaginary example a designer is investigating ways to develop some kind of large structure. Traditional approaches come to mind: trusses, cable structures, etc.

At one point the designer decides to experiment with structural 'rules' that could be used in a repetitive way. This is where L-systems can come handy. The first idea that comes to mind is the possibility of linking some type of fractal with a triangulated approach to add rigidity to the structure. Using the L-system Java Applet one could easily generate something like this:



```
Redraw
number of cycles: 4
rotation angle (R): 60.0
examples: Example 3

Alphabet:
  F: forward, draw line
  f: forward, no line
  +: turn right by angle R
  -: turn left by angle R
  |: turn 180 degrees
  [: start a branch
  ]: end branch

Other: current state
         preserved

Axiom: FXF--FF--FF
Production Rules:
F -> FF
X -> --FXF++FXF++FXF--
Rule: F -> F-F++F-F
```

*Figure 6. Java Applet*

The production rules consist of only two lines. At this point one could a) proceed and generate a flat structure that could be later modified, or b) develop a new set of three-dimensional rules. In this particular case it was decided to generate a new set of rules ending with what is called a three dimensional Sierpinski gasket. The CAD visualization looks like this:

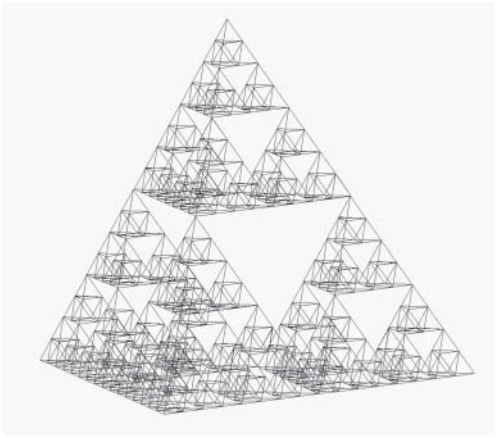


*Figure 7. Sierpinsky Gasket Form Generated by the Java Applet.*

This particular form has nine levels of recursion. The designer can simply change the level of recursion to a higher or lower number to achieve immediate changes. She or he can also easily modify other rules to achieve other forms in a second. If one were to generate this particular structure from the very beginning, the effort would certainly take some time. Here, the example only took a few minutes testing the geometry with the assist of the JAVA applet. Assuming that this is an interesting point of departure that merits further exploration, the CAD file is then read by NURB modeling applications. In this example, the designer has chosen to only take a section.

This basic model could then be stripped in order to generate a simple wireframe model where the individual bars would be subject to NURB deformation. The individual bars would then be given the necessary thickness as shown here:

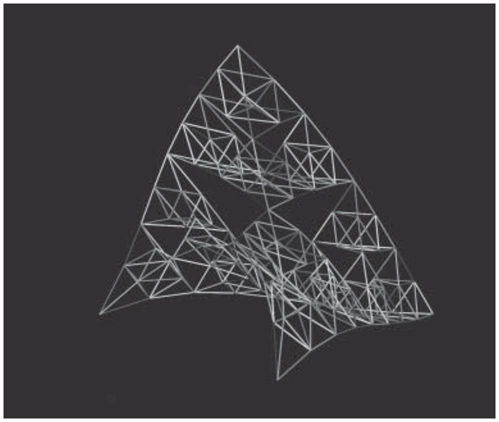


*Figure 8. Deformation of the basic form using NURB application.*

As part of the design process, the component could then be assembled along with other parts of the building proposal to test its validity and aesthetic qualities. A preliminary rendering would look like this:

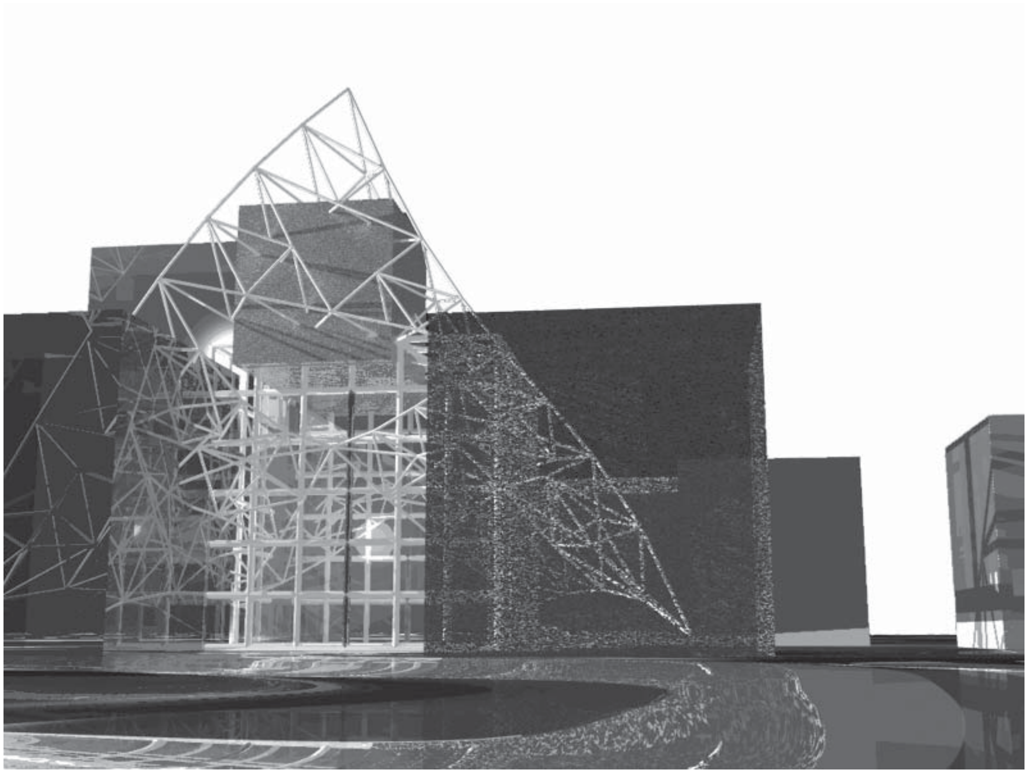


*Figure 9. Final result included in the architectural composition.*

Like in the first example illustrated above, the designer can certainly adopt the scheme, but can also test other alternatives by simply modifying the re-writing rules. The process only took a few minutes.

## Conclusions

Current computer modeling applications are OK. However, they need to offer additional options that give designers more form generating capabilities other than the rather tedious current approach involving the point-line-surface paradigm. This is where L-systems come into play. By using L-systems and migrating their results to any CAD and/or NURB application, designers can not only discover new forms and arrangements, but can considerably simplify the way geometry is constructed.

The experiments presented in this paper are only a very small sample of the total output generated. They illustrate great design potential. The success of the project can be evaluated by how easy it was to conceive, develop and visualize potential approaches to design solutions. From the writing of the six rules needed to generate the basic components of the geometry to the final rendering stages, it only took a few minutes in terms of issuing commands. There were no complicated series of steps taken to arrive at the final design of components. No effort was required at solving complex intersections.

The bottom line was that instead of learning and applying some obscure software command or instruction, actions that take a substantial amount of time and energy to understand and apply, the effort was directed at testing a variety of design propositions that dealt with scale, proportions, placement of elements, colors, tectonic qualities, and all the other cool functions that make architectural design so much fun to play with.

## References

Prusinkiewicz P, and Lindenmayer A. (1990) The Algorithmic Beauty of Plants. Springer-Verlag.

Prusinkiewicz P, Hanan J.S, Mech R. (2000). An L-system-based plant modeling language. In: Lecture Notes in Computer Science 1779: Applications of graph transformation with industrial relevance. (Nagl M, Schurr A, Munch M, eds.) Berlin: Springer-Verlag, 395-410.

Schneider, C.W. & Walde, R.E. (1992). L-system computer simulations of branching divergence in some dorsiventral members of the tribe Polysiphonieae (Rhodomelaceae, Rhodophyta). Phycologia, 31: 581 - 590.

Sschneider, C.W., Walde R.E. & Morrelli, R.A. (1994). L-systems computer models generating distichous from spiral organization in the Dasyaceae (Ceramiales, Rhodophyta). Eur. J. Phycol., 29: 165-170.

Coen, E., Rolland-Lagan A., Matthews M., Bangham A., and Prusinkiewicz P., The genetics of geometry. Proceedings of the National Academy of Sciences 101 (14), pp. 4728-4735.

Prusinkiewicz P., Art and science for life: Designing and growing virtual plants with L-systems. Acta Horticulturae 630, pp. 15-28.