GENERATING 3D PLANTS USING LINDENMAYER SYSTEM

Kamil Ciosek and Pawel Kotowski
Faculty of Mathematics and Information Science, Warsaw University of Technology
Pl. Politechniki 1, 00-661 Warszawa, Poland

Keywords: Modeling, Lindenmayer system.

Abstract: The main focus of this paper is the analysis of modern methods of algorithmic plant generation. First, a brief introduction is given to the necessary formalism: Lindenmayer system. It is followed by a description of each stage of plant generation process. These include algorithms for obtaining: leaf venation graph, leaf texture, stem texture, and the geometry and topology of the whole plant. In particular, the following approaches have been used: textures are obtained from transformed noise, a general plant description is generated with a parametric Lindenmayer system and a purpose-built particle-based algorithm is used to simulate leaf venation. The last section gives a detailed description of four sample systems used to generate different plants, outlining the reasons why a given system gives the desired graphical result.

1 INTRODUCTION

It is beyond doubt that plant modeling is one of the milestones that computer graphics needs to achieve to finally get to the holy grail of movie-quality real-time image synthesis for games and other virtual worlds. Indeed, after programmers mastered the ways to reproduce simpler elements of our world on the computer screen, it is now primarily in the foliage that the struggle goes on to stun the user with the realism of artificial imagery. While each new generation of games presents a considerable improvement over its predecessors, the results are still not entirely satisfying and usually apply to only a specific class of plants. Therefore plant generation truly stands out as an area that is worthwhile to research into and learn about.

With most objects, realistic display boils down to hand-crafting a model with a polygon count sufficient to display the necessary detail. This approach has quite limited use because no model designer could possibly reproduce the level of complexity represented by a plant, if the model is to be looked at from a close enough distance. Also, it is impossible to achieve the variety of plants within the same species, or trace the development of plants over time with manual modeling. Therefore we need a descriptive formalism that enables us to compress the plant structure into a workable formula that can be hand-modified and intuitively understood. In fact, it is one of the central concepts of mathematics to describe some aspect of the structure of a complex object in a simple fashion. One feature of plants that seems to be helpful in doing so is the fact that they display a certain degree of self-similarity. Of course, this fractal-like behavior does not encompass every aspect of the plant at every scale, but still, it is quite helpful. In this respect, this paper describes the concept of Lindenmayer systems: a type of formal grammars that is especially wellsuited for the modeling of plants and that allows the user to easily exploit whatever self-similarity the plant has.

The following chapters give an account of experiments with Lindenmayer systems, as well as of efforts to patch them up by adding external algorithms to model the aspects of the plant that they poorly represent (most notably, leaf venation patterns).

2 LINDENMAYER SYSTEMS

This section introduces the concept of Lindenmayer systems, a variation of formal grammars widely applied to plant generation. They were first introduced by Aristid Lindenmayer as a means of modeling the growth of algae, but were later given a more thorough theoretical description and applied to many different problems.

Ciosek K. and Kotowski P. (2009).
GENERATING 3D PLANTS USING LINDENMAYER SYSTEM.
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications*, pages 76-81
DOI: 10.5220/0001785300760081
Copyright © SciTePress

A deterministic context free Lindenmayer system (D0L-system) is a tuple $\langle V, \omega, P \rangle$ where

- V - is a set of symbols.
- $\omega \in V$ - is the axiom or start symbol.
- $P \subset V \times V^*$: $(\forall a)(\exists! p)((a, p) \in P)$ is the set of productions.

Note that the right side of the production may be empty (erasable productions are permissible) and that Lindenmayer systems do not differentiate between terminal or non-terminal symbols. Also note that each symbol is the left side of exactly one production (hence the determinism).

The arrow relation $\rightarrow \subset V^+ \times V^*$ is defined as follows: $P \rightarrow Q$ iff P can be expressed as a sequence of symbols $P = p_1, p_2, ..., p_n : p_i \in V$ and Q can be expressed as a sequence of strings $Q = q_1, q_2, ..., q_n \in V^*$ such that $n > 0$ and for each i there exists a production from p_i to q_i, i.e. $(\forall i)(p_i, q_i) \in P$.

The difference between ordinary grammars and Lindenmayer systems lies in the fact that we substitute all symbols at once. Also note that, given a string from V^*, the concatenation of the transformations of each constituent letter of the string is defined unambiguously and is trivial to compute. This is in strict opposition to context-free grammars, where special care needs to be taken to avoid ambiguity.

In practical applications, we start off with the axiom and iterate the $\rightarrow$ relation a fixed number of steps. The number of steps is considered a parameter. Normally, this parameter represents the ”depth” of simulation, for example the level of development of the plant being modeled. Below, a simple example of a deterministic context-free Lindenmayer system is presented. A more elaborate example, which can be used to model plants will be given later.

$V = \{A,B,C\}$
$\omega = A$

The set P contains the following productions:

$A \rightarrow BC$
$B \rightarrow AC$

Note that we have omitted the production from C. This is common practice, and means that $C \rightarrow C$. After three iterations, this system yields:

$\omega = A \rightarrow BC \rightarrow ACC \rightarrow BCCC$

It has been decided that context-free parametric Lindenmayer systems provide enough flexibility to model an adequate scope of plants. The plant is constructed in that the string resulting from iterating a Lindenmayer system a specified number of times is scanned for the symbols listed below (all other symbols are ignored).

• F(l, r)	Draws a stem segment(cylinder) of length l and radius r. The cylinder follows the Z axis. Its base matches the XY plane.
• L(l)	Draws a leaf. The length of the main axis of the leaf equals l.
• [	Puts current turtle state onto the stack.
•]	Discards current turtle state and pops the new state
• +(α)	Rotates the turtle by α degrees around the X axis.
• &(β)	Rotates the turtle by β degrees around the Y axis.
• /(γ)	Rotates the turtle by γ degrees around the Z axis.

The drawing takes place in a turtle-like manner in that the drawing turtle has a state at any given time. The state comprises location and rotation (which is represented as a 4×4 transformation matrix). Therefore, turtle state can be viewed as an alternative reference frame embedded into the 3d scene.

3 PLANT GENERATION

This chapter addresses the core issues related to plant generation. While the whole process revolves around Lindenmayer systems, auxiliary mechanisms need to be added to make the plants appear realistic. In the following subsections, a discussion is given of the methods used for the generation of various aspects of plants.

3.1 Leaf Venation

Modeling leaf venation is quite imperative to achieving the proper looks of a modeled plant. This is due to the fact that the surface of plant leaves is usually bigger and more prominent to the viewer than other features of the plant. Therefore due care must be exercised to ensure that an adequate algorithm is employed.

Taulor-Hell and Baranoski (2002) provide a thorough list of methods used to date, ranging from simple texture mapping of scanned leaves to modern procedural approaches.

Couder et al. (2002) describe an interesting experiment aimed at establishing new methods of reproducing venation patterns. They put special gel in moulds of different shapes. The gel was then left to dry and cracks that appeared on the surface due to the stress caused by the top layer of the gel drying faster and than the lower layer. They concluded that the pattern of cracks on the surface of the gel bears considerable similarity to venation patters of various leaves.

Rodkaew et al. (2002) describe a particle-system based approach that is very appealing. It has, however, one major disadvantage: it is difficult to tweak the used approach to achieve true similarity with real leafs.

Runions et al. (2005) provide a solution that is quite successful at addressing the ills of the earlier attempts of using particle systems to model vein growth. The authors, base their algorithm on the idea that venation patterns emerge due to a special plant hormone, called auxin: “[...] auxin originates in the leaf blade and flows toward existing veins, which transport it to the leaf base. During this flow, auxin is canalized into narrow paths [...]. These paths gradually differentiate into new vein segments. Experimental evidence suggests that auxin sources may be discrete.”

To account for this hormone, we use two kinds of particles: source particles, representing auxin and node particles, representing vein segments. The algorithm is an iterative process that tries to reproduce the way in which the sources and nodes relate to one another on the leaf surface.

Once the iterative process has completed, we have a graph representing the generated venation pattern. What we would like to have, however, is the texture of the actual veins. To get it, we need to account for one vastly important aspect of leaf venation: the width of the veins. First, we observe that the venation pattern of leaf generated using the described algorithm has the topology of a tree. This means that the edges (vein segments) may be sorted with respect to their distance from the tree root (leaf origin). Therefore, it is possible to mark the edges which are the furthest from the root (the thinnest veins) as having width 1. The width of the remaining edges can then be assumed to follow from the formula $r_{parent}^{n} = \sum r_{child_i}^{n}$. We calculate the widths of edges that are farther from the root first and then use these values to calculate the nearer ones. The process is repeated till we reach the root node.

3.2 Leaf Texture

Once the geometry and topology of the leaf venation pattern have been generated, there still remains the question of how to texture the areas of the plant leaf between the veins. Ideally, the color of the leaf surface in these areas should be dependent on the surrounding vein pattern and on the shape of the leaf. However, it turns out that satisfactory results may be obtained by using a simple technique of making the background entirely independent of the venation pattern.

The following figure compares the actual photo of a croton leaf with an artificially generated equivalent. As can be seen, considerable dissimilarities remain apparent.

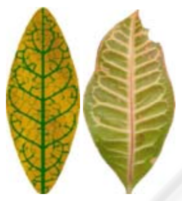

Figure 1: Generated leaf vs. real leaf image (source: own photo).

3.3 Triangulation of Leaf Surface and Deformation into the 3D

It is of course imperative to achieving the proper looks of the plant that they are wrapped in a natural fashion. The best, most general solution, is the one employed by Mundermann et al. (2003). They have used sticky splines, a modification of spline curves that maintains the topology of the modeled structure, to construct the leaf skeleton and devised a special algorithm that is able to generate such skeletons automatically. Naturally, the constructed skeleton corresponds directly to the leaf’s venation pattern. This allows them to represent arbitrary leaf lobes and thus produce high-quality renderings of the plants. This approach, however, is both complicated and computationally expensive, since each primary or secondary leaf vein must be given its representation in the form of an appropriate spline. Therefore, it is better suited to plant rendering than to real-time display.

In search of a simpler solution, it became apparent that as long as we do not need to model leaf deformation that is due to the venation pattern, it suffices to provide just an arbitrary triangulation of a flat leaf shape, which can then be bent into the third dimension. It is, however, important that the triangulation is accurate enough near the brim of the leaf blade, so that jagged edges can be avoided.

Once we have the mesh, it has to be deformed to account for the bending of the leaf. The algorithm uses a simple, but relatively effective solution that assumes that the deformation of the leaf surface into the third dimension is a function of only the y coordinate of the flat leaf surface (the one that goes along the leaf axis).

3.4 Stem Texture

While there seems to have been considerable research into the ways of generating aesthetically appealing textures of tree bark, we could not find a ready algorithm specifically geared at reproducing the outer looks of the stems of non-tree plants. Because devising a new algorithm would be complex, we have opted to use a simple bark-like pattern that is fast and easy to compute while delivering results of passable quality.

We adopted the solution due to Oppenheimer (1986) which used a noise pattern run through a sawtooth function. The algorithm takes three inputs: a noise image, an integer N specifying the number of bark ridges and a real R specifying the roughness of the bark.

The result of the described procedure is a grayscale image. To obtain color, the image is saturated using a gradient specified by two user-definable colors.

The following figures demonstrate textures generated using various parameters

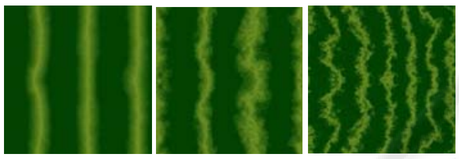
Figure 2: Samples of stem textures generated by the described algorithm.

4 SAMPLE PLANTS

The first system, heavily adapted from Prusinkiewicz and Lindenmayer (1996), has been used to generate a generic plant.

```
axiom  → A(100,5,200)
A(s,t,l) → [&(22.5)F(s,t,l)L(l)A(s,t/2 + 2,l)] /(5*22.5)
           [&(22.5)F(s,t,l)L(l)A(s,t/2 + 2,l)] /(7*22.5)
           [&(22.5)F(s,t,l)L(l)A(s,t/2 + 2,l)]
F(s,t,l) → S(t,s,l) /(5*22.5) F(s,t,l)
S(s,t,l) → F(t,s,l)
```

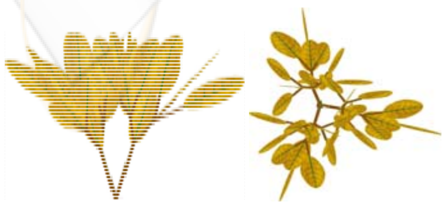
Figure 3: Plant generated from the system above (3 iterations).

The way the system works is centered around the A token. This token has three parameters, which stand for the following: the length of a single stem segment, the width of the stem segment and the length of the leaf. In each iteration, the A token produces three branches using the [and] stack operators. They protrude from their base at different angles (the / operator). Each branch consists of an appropriately rotated (&) stem segment (F), a leaf (L) and the token A which allows it to split further and form child branches in subsequent iterations. Note how the stem width is decreased as the plant grows. It is guaranteed to be at least 2 so that the stem remains visible.

Below is presented a second system that reproduces a yucca plant

```
axiom                → A(175,25,250)
A(s,t,l)             → F(s,t) B(7,6,l,60,s,t)
B(h,v,l,d,s,t)       → BN(h,v,v,l,d,s,t)
BN(h,v,i,l,d,s,t)    → LR(h,l,d*v/i) /(13) F(s/10,t)
                       BN(h-0.4,v-1,i,l,d,s/2,t)
LR(n,l,d)            → LRN(n,n,l,d)
LRN(i,n,l,d): i = 0 → eps
LRN(i,n,l,d): i > 0 → [+(d)L(l)] /(360/n) LRN(i-1,n,l,d)
```

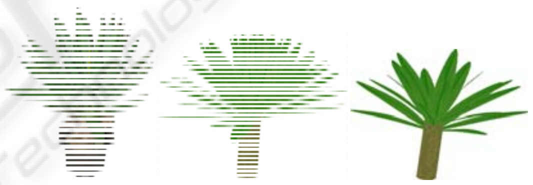
Figure 4: A real yucca photo (source: www.pyraflora.co.za) together with models generated with 100 (centre) and 10 (right) iterations of the system above.

This model was been inspired by a photo of a real yucca plant. First, look at the last three productions below: they represent a ”procedure” ”invoked” by using the LR(n,l,d) token. It draws n uniformly distributed leaves of length n and inclination d. The two productions from B and BN generate a number of such concentric leaf groups (once group is added for one iteration). The parameters of the leafs are varied to achieve a less symmetrical look. In particular the inclination change reproduces the dome-like shape of the whole plant. After each group of leaves has been drawn, the coordinate system is rotated so that the leaves of the next group do not protrude from the plant at the same angles. Also, a short stem segment is added. The token A is used to add the first, long stem segment and initiate the generation process.

The third system reproduces a fern leaf. It is a heavily modified version of a system proposed by Prusinkiewicz and Lindenmayer (1996).

```
axiom        →  S
S            →  F(2,1) [ +(40) /(90) L(11,1) ]
                [+(-40) /(90) L(11,1) ]
                +(9) F(2,1) [ /(90) L(11,4) ]
F(s,t)       →  F(s*1.2,t)
L(l,i): i < 4 →  L(l*1.2, i+1)
L(l,i): i = 4 →  [ / (270) S ]
```

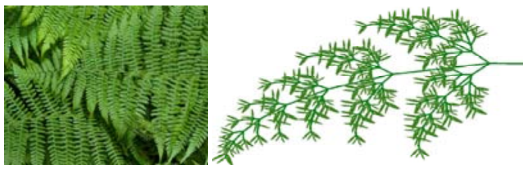
Figure 5: A real fern photo (source: Wikipedia) together with a generated model (20 iterations).

The basic idea of this system is based on the fractal-like structure of a fern leaf, where smaller elements have the same structure as larger elements. The basic building block of the fern leaf is represented with the production from S: two branches (L) protrude from a stem made from to segments. The rotation before drawing the leaves is necessary so that the surface of the leaves is aligned with the surface of the whole leaf. The rotation +(9) before the creation of the second stem segment is used to make the whole leaf bend. Note how the L symbol, whose main purpose is the creation of leaves is used to create branches. Its second parameter is used in a timer, which converts the leaf to the basic building block after a fixed number of iterations (this is done using the two last productions). This way, the youngest generation of created objects is rendered in the form of leaves. The disadvantage is that this has absolutely no biological motivation, but it looks good enough. The production from F is used to elongate the existing stem segments so there is enough place for the emergence of new ones in the following iteration steps.

The next system reproduces a cabbage head.

```
axiom        →  B(4,100,75)
B(h,l,d)     →  LR(h,l,d) /(13) B(h-0.3,l,d/2)
LR(n,l,d)    →  LRN(n,n,l,d)
LRN(i,n,l,d): i = 0  →  eps
LRN(i,n,l,d): i > 0  →  [+(d)L(l)] /(360/n)
                        LRN(i-1,n,l,d)
```

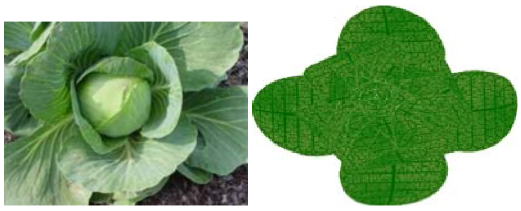
Figure 6: A real cabbage photo (source: www.hort.purdue.edu) together with a generated model (10 iterations).

The idea of the system arose when working on the yucca plant, and indeed the two systems are similar, and the leaf-drawing procedure represented by the symbol LR is even identical. This is an excellent example of how a seemingly small modification to the system yields a completely different plant (although other aspects of the plant have been modified as well). The key difference is in the way the inclination of the leaves in controlled: with the yucca plant, the inclination changed linearly with respect to the iteration step, here it decreases exponentially (the d/2 parameter in the second production). This has the effect that the concentration of leaves near the plant centre is much higher than on the boundary. Also, the leaves near the plant centre begin to self-intersect, which of course is not realistic as such, but creates a visually pleasing filled area near the centre of the plant.

5 CONCLUDING

It is quite obvious that the issue central to the whole process is the question of how to get the utmost use from the formalism of Lindenmayer systems. Sadly, it seems adequate to conclude that the promised biologically-motivated means of modeling organic structures in a fast and easy way has yet to come into being. The problem with Lindenmayer systems is inherently tied to one of their main virtues: simplicity. True, it is possible to express complex structures using only a few productions. True, it is easy to obtain images of the same plant at different developmental stages if the system is appropriately constructed. This does not change the fact, however, that it is extremely difficult to extend this formalism to cover a broad spectrum of objects. Lindenmayer systems are good at describing tree structures, which is hardly surprising because trees are simple. As soon as more complexity is required, they fail. One may argue that most plants do have a tree topology and thus the added complexity is not required. The simplest example of this limitation it the one that has been encountered while researching leaf venation: originally, the venation pattern of the leaf was intended to be modeled using an auxiliary Lindenmayer system. However, it proved impossible to construct a system that would model a reasonable variety of such systems adequately as it was very difficult to make the separate vein lets grow together. Consequently, a separate algorithm had to be introduced. Plants may be tree like in the macro scale, but they are certainly not so in the micro scale, nor in the scale of the whole ecosystem. A similar argument applies to other plant features. If one wants smooth branches, it is necessary to add a generalized cylinders to the model, which is external to the system. If one wants flowers, another structure has to be added. This has the effect that once we add everything that is necessary to construct a well-looking plant, the whole model loses its flexibility because these addenda do not have the developmental potential that a raw Lindenmayer system boasts: we can no longer trace the way a plant develops. Indeed, during the development of productions, one is fast tempted to fall into the pitfall of merely viewing the Lindenmayer system as an exotic variation of programming in LOGO and thus lose whatever biological founding the model might have had. Naturally, this does not mean that Lindenmayer systems are out of place. As of now, there exists no better solution for generating arbitrary plants.

Another aspect of plant modeling that needs to be stressed here is the huge potential of particle systems. In the effort described in this paper, they have been used to model the leaf venation pattern. Their main advantage is the relatively straightforward way of implementation, at least compared to attempts to tackle the same issues using a more prescriptive approach. It is also easy to introduce variation in the generated structures, because the sources are scattered randomly as well as to model two- or even three-dimensional structures. Actually, attempts have been made (Rodkaew et al., 2002) to use them for modeling whole plants, but initial results were modest at best. In this context, it seems appropriate to note the analogy between particle and Lindenmayer systems: if we allow the particles to have arbitrary parameters and the rules that govern the behaviour of a particle (which may mean both modifying an attribute of the particle or splitting it into smaller particles) to be based on an arbitrarily defined neighborhood of the particle (which may extend to the whole system), then a Lindenmayer system is just a special case of a particle system constrained to one dimension and one notion of proximity, where tokens correspond to particles. It would be interesting to see in what practical ways the use of particle systems may be beneficial to the modeling of plants.

In summary, it does not seem very original or innovative, but needs to be stated that plants are inherently complex. Complex objects require complex models, which usually require complex implementation. This paper outlined some endeavors on the way to a better model. It remains to be seen how fast the evolution of computer graphics leads us to an algorithm that produces truly satisfying results.

REFERENCES

Couder, Y., Pauchard, L., Allain, C., Adda-Bedia, M., Douady, S., 2002. The leaf venation as formed in a tensorial field. *The European Physical Journal B - Condensed Matter and Complex Systems*, 28(2):135–138. ISSN 1434-6036.

Mundermann, L., MacMurchy, P., Pivovarov, J., Prusinkiewicz, P., 2003. Modeling lobed leaves. cgi, 00:60, ISSN 1530-1052.

Oppenheimer, P.E., 1986. Real time design and animation of fractal plants and trees. *SIGGRAPH Comput. Graph.*, 20(4):55–64. ISSN 0097-8930.

Prusinkiewicz, P., Lindenmayer, A., 1996. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, USA. ISBN 0-387-94676-4.

Rodkaew, Y., Chongstitvatana, P., Siripant, S., Lursinsap, C., 2002. An algorithm for generating vein images for realistic modeling of a leaf. In *International Conference on Computational Mathematics and Modeling*. May, Thailand.

Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A-G., Prusinkiewicz, P., 2005. Modeling and visualization of leaf venation patterns. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 702–711, New York, NY, USA.

Taulor-Hell, J., Baranoski, G., 2002. State of the art in the realistic simulation of plant leaf venation systems. *Technical Report CS-2002-17*, University of Waterloo, Department of Computer Science, Waterloo, Ontario.