Plants, fractals, and formal languages

Article *in* ACM SIGGRAPH Computer Graphics · July 1984

DOI: 10.1145/964965.808571

1 author:

Alvy Ray Smith
Independent Researcher
53 PUBLICATIONS **2,749** CITATIONS

SEE PROFILE

PLANTS, FRACTALS, AND FORMAL LANGUAGES

Alvy Ray Smith.

Computer Graphics Project
Computer Division
Lucasfilm Ltd.

ABSTRACT. Although fractal models of natural phenomena have received much attention recently, there are other models of complex natural objects which have been around longer in Computer Imagery but are not widely known. These are procedural models of plants and trees. An interesting class of these models is presented here which handles plant growth, sports an efficient data representation, and has a high "database amplification" factor. It is based on an extension of the well-known formal languages of symbol strings to the lesser-known formal languages of labeled graphs. It is so tempting to describe these plant models as "fractal" that the similarities of this class of models with fractal models are explored in an attempt at rapprochement. The models are not fractal so the common parts of fractal theory and plant theory are abstracted to form a class of objects, the *graftals*. This class may prove to be of great interest to the future of Computer Imagery. Determinism is shown to provide adequate complexity, whereas randomness is only convenient and often inefficient. Finally, a nonfractal, nongraftal family of trees by Bill Reeves is introduced to emphasize some of the paper's nongrammatical themes.

CR CATEGORIES AND SUBJECT DESCRIPTORS: F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages - *Classes defined by grammars or automata*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism; J.5 [**Arts and Humanities**]: *Arts, fine and performing.*

ADDITIONAL KEY WORDS AND PHRASES: plant, tree, graftal, fractal, particle system, parallel graph grammar, L-system, database amplification, Computer Imagery.

INTRODUCTION

Two of the newest contributions to Computer Imagery are the adaptive, "fractal" technique of Fournier, Fussel, and Carpenter‡[7], and the particle systems of Reeves[19]. Both of these are displayed in the *Genesis Demo*[20], the growing mountains realized by fractals and the fires by a particle system. Another example is the picture *Road to Point Reyes*[2], the mountains again being fractal and the grasses particulate. They are both departures from traditional computer graphics which takes the "cubist" approach of constructing models from geometric primitives, now the domain of CAD/CAM. *Computer Imagery* is used to refer to the newer, more flexible and subtle state of the art of computed pictures.

A main purpose of this paper is to present another class of complex renderable objects. Members of this rich class will be interpreted as plants or trees. The class shares characteristics with fractals and particle systems; another purpose of the paper is to make the relationships clear. The definitions from Benoit Mandelbrot's inspiring book[14] apparently do not allow the plants presented here to be described as fractals (see his Chapter 16 in particular) because the notion of fractal is strongly geometrical and defined only in the limit. What may prove to be of as much or more interest is structural rather than geometrical and becomes sufficiently intriguing far below the limit. This paper is concerned with formal language techniques, instances of which implicitly underlie many of Mandelbrot's examples and, as will be shown below, other Computer Imagery techniques.

The key idea is contained in the nature of formal languages. The plants presented are words in a formal language - in particular, a *parallel graph grammar*[13] language. These languages are not the well-known Chomsky hierarchy[10] languages but the lesser known *L-systems*[9,12]. It is suggested that the parallel graph grammar languages - the *graftals* - have more potential for Computer Imagery than fractals because they are less restrictive.

Graftals share with fractals the attribute of "the closer you get, the more it looks the same" when a scaling geometry is imposed. Fractals have been explored by investigating the special cases of strict and statistical self-

‡ But it is the *locally* adaptive technique of Carpenter which is emphasized here.

similarity which permit actual computation of the Hausdorf-Bescovitch dimension. This is of only limited usefulness in Computer Imagery. What will be shown to be more useful is a relaxed "self-similarity" which is an ability to generate detail preserving the nature of an object without strictly copying it. This attribute is forced by the finiteness of a formal grammar. "Fractality" is thus a special characteristic achieved by only some graftals. Those fractals generated with true random processes - e.g., fractional Brown fractals - are not graftals. Although the random fractals can be used to generate beautiful still pictures[14,23] their computational practicality is yet to be demonstrated, particularly in the context of animated sequences.

An advantage of the graftal approach is its intrinsic locality. Locality admits adaptive subdivision which has proved valuable in Computer Imagery for its computational savings in patch rendering and spline computation.

An implication of the finiteness of grammars is that randomness is not allowed. Surprisingly, evidence will support the acceptability of this restriction. In fact, it is computationally more efficient to avoid random number generators when objects as complex as plants are rendered.

In addition to the plant models which are the primary focus of this paper, several other models are discussed including two grammars from Mandelbrot, mountains by Carpenter, and particle system trees by Reeves in order to elucidate the functions of locality, randomness, and formal grammars in systems designed to generate complex images from small databases, a property called *database amplification*.

ALGORITHMIC PLANTS

Lindenmayer[12] introduced the notion of parallel rewriting grammars for the modeling of developing biological systems. The grammars are similar to those of conventional formal language theory[10] except productions are applied simultaneously, and there is no distinction between terminal and non-terminal symbols. All strings generated from an initial string, or axiom, are considered to be words in the language of the grammar. The axiom is typically an element of the grammar's alphabet, but this is not a requirement here. An extensive literature[8] has developed on these so-called *L-systems,* particularly for the context-free subset (the *0L-systems,* for zero neighbors) and for the two simplest context-sensitive subsets (the *1L-systems,* for a one-symbol, nearest-neighbor context and the *2L-systems,* for the two-symbol, nearest-neighbors context).

Lindenmayer also introduced the notion of *bracketed* L-systems which extend an L-system alphabet by the set {[,]}. This allows the representation of trees with strings. The brackets contain branches which are attached at the symbol just left of the left bracket. A simple example is the (context-free) $0L$-system with alphabet {0, 1, [,]}, axiom 0, and production rules {0→1[0]1[0]0, 1→11, [→[,]→]}. The first three steps (generations) are 0, 1[0]1[0]0, and 11[1[0]1[0]0]11[1[0]1[0]0]1[0]1[0]0. An equivalent graphic presentation of the system above is presented in Figure

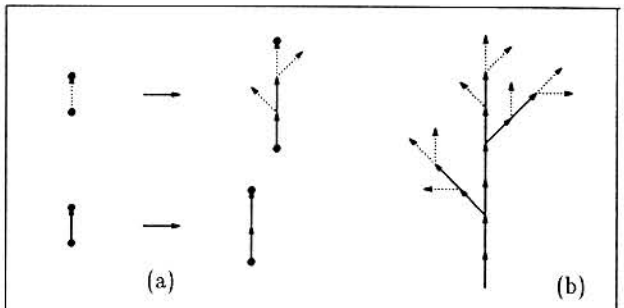

Figure FREETREE. (a) Production rules. (b) Generation $n = 2$.

FREETREE. Notice that affine transformations of the graphical statements of the rules are required and that dots are used to show corresponding connections before and after replacements. Notice also that two kinds of branching, left and right, have been used rather than just the one of the string representation. Another set of bracket symbols could represent this extension; so the branching rule becomes 0→1[0]1(0)0. Representing diversity with larger alphabets will receive further mention when the role of randomness is discussed.

The (geometric) trees generated are considered to be data structure maps, not necessarily the final image. A postprocessing step, called an *interpretation,* expands this map, assumed to have only a finite amount of information at each node (finite alphabet), into the final image. Thus all trees structurally equivalent to the data structure tree under finite interpretation can be considered to be words generated by the grammar. Notice that finite interpretation does not allow randomness.

Notice also that the string representation of these plants is quite concise. Any language - e.g., C - which has character or byte manipulation operators permits easy, speedy implementation.

Finally, notice that if $T(n)$ is the n-th generation, then the $(n+1)$-th generation can be expressed recursively in string notation as

$$T(n+1) = 1^{2^n}[T(n)]1^{2^n}[T(n)]T(n) .$$

In the terminology of Mandelbrot ([14] pp. 152-3), the tree is a *subfractal* (seems like a fractal but isn't) which is not *self-similar* but has a *residue* which corresponds to the progeny of the 1's in the trunk of generation $n=1$. In formal language terminology, the residue is the consequence of another symbol in the alphabet. In the example above, the production rule for the solid arrow is not very interesting, so "residue" is perhaps appropriate, but in general its rule could be as interesting as that of any other symbol.

Plate CARTOON.TREE is a 2-dimensional representation of the 7th generation of this system. Since the string representation of a plant is referred to as its *gene,* or *genotype,* it follows that the plant is the *phenotype* of the string. The phenotype in this example is obtained from the genotype by representing each 0 or 1 with an antialiased line segment. At the end of each branch - thus at each] in the gene - a "leaf" is drawn which, in this simple case, is just an antialiased disk.

The program used, called **Gene**, gives control over color of stem and leaf, provides a dropped shadow for each primitive, does depth darkening, and provides separate width controls for stems and leaves. It also generates a 3-dimensional database for full rendering with hidden surfaces removed; the 2-dimensional versions are simply speedy sketches and color studies. There is no reason the leaves or stems could not be more complex, but the simple shapes used so far - lines and disks in 2-D or cylinders and spheres in 3-D - have yielded surprisingly pleasing results (Plate WHITE.SANDS). Notice that the branches get smaller with distance from the root and the leaves get larger. These attributes are not part of the grammar but are variations modeling global gravitational, chemical, or electromagnetic tropisms. They are added during interpretation of the word generated by the grammar. Kawaguchi[11] gives many examples of the beauty which can be obtained at the interpretation step of a simply generated word.

The program **Gene** also allows the use of an arbitrarily large (or small) finite set of angles or a random set if so desired. It will be demonstrated that randomness is not required for pleasing results, but just one angle and its negative do not suffice, as demonstrated by Plate CARTOON.TREE. The addition of more complexity in the form of context is studied next.

Hogeweg and Hesper[9] studied the *propagating, deterministic* bracketed 2*L*-systems, where "propagating" means there are no erasing rules and "deterministic" means no two distinct rules share the same left side. With these systems they obtained a wide variety of plant-like species. Their paper, plus the books of Stevens[22] and Mandelbrot[14], are the principal inspirations for my work here. Hogeweg and Hesper were restricted to simple black-and-white line drawings and only 25 generations (successive applications of all applicable productions). I have been able to apply full-color 3-D Computer Imagery techniques to their results, add flowers and leaves, and use much deeper generation trees (35-45 generations typically suffice) to make stills, growth movies, and a hologram.

An example of a bracketed 2*L*-system has the alphabet {0, 1, [,]}, the axiom 1 and the production rules 0.1[1].1.1.0.11.1.0, where the dots separate the images of the eight binary triples in numeric order, left to right. Thus 001 maps to 1[1], meaning that wherever a 0 occurs in a string with a 0 to its left and a 1 to its right, it is replaced at the next step (generation) with the string 1[1]. There are assumed to be invisible 1's at the unattached ends of all branches. Only the 0's and 1's are replaced, the brackets being structural markers only. The left neighbor of a branch is the 0 or 1 just left of the branch - e.g., in the string 0[1], the left and right neighbors of the 0 are two invisible 1's and those of the 1 are the 0 and another invisible 1, respectively. The first 11 generations of the bracketed 2*L*-system above are shown in Table 1. An equivalent description of the system is graphical, Figure SENSTREE. The left-side symbol to be replaced by the right side of a rule is surrounded by a dashed box. All productions not shown leave a symbol unchanged. From this inauspicious beginning arises - after 35 generations and less restrictive branching - the plant of Plate WITH.WITHOUT which shows the same plant, in two dimensions, with and without flowers to reveal its beautifully intricate branching structure.

n	L(n)
0	1
1	0
2	11
3	00
4	01[1]
5	111[0]
6	000[11]
7	001[1][10]
8	01[1]1[0][111]
9	111[0]0[11][000]
10	001[11]11[10][001[1]]

Table 1. 11 generations of the 2*L*-system 0.1[1].1.1.0.11.1.0.

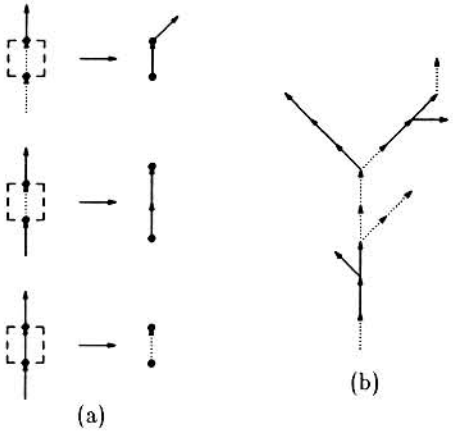

Figure SENSTREE. (a) Production rules. (b) Generation $n = 11$.

By rendering successive generations in successive frames, a plant's growth can be animated. See Plate WISP.GROWTH. To ensure coherence between successive generations, an angle must be "nailed" to its branch once it has been selected. Plates GARDEN, GARDEN.DROP and BUSHES demonstrate a variety of related graftals with and without flowers or leaves. Plates GREEN.FLAME and VITA.PLANTS illustrate application of **Gene**'s controlling parameters.

If a geometry is forced on a plant grammar, so that a production rule requires the same space be occupied before and after replacement, then the grammar has the property that "the closer you get, the more it looks the same". Thus, as we zoom in on a plant generated by such a grammar, we cause further invocation of the replacement rules to generate more detail. This detail will resemble the overall plant since it is generated by the same small set of rules. So the plants have a form of "self-similarity" which is much looser than that associated with fractals.

CARPENTER MOUNTAINS

Loren Carpenter has shown that a grammar with the rule shown in Figure CARPENTER suffices to turn a database (axiom) of a small set of triangles into a rich mountain at the 10th generation, more or less. Plate FOUR.FRACTAL is the 5th generation in a Carpenter grammar with one triangle as the axiom. Clearly, database amplification is one of the benefits of this grammar. We have apparently been quite lax in applying the production rule of the Carpenter grammar since in no case is its left side literally (geometrically) satisfied even allowing affine transformations. Our formal language must allow topological deformation in the sense that any deformation required of a left side to make it apply must be applied as well to the right side before replacement. The rule must apply to any triangle.

There is a problem with shared edges here. How is the common edge between two triangles to be replaced identically as the result of two separate applications of the production rule, each requiring a different deformation? There is no way to solve this problem in the style of this paper if any kind of infinity is involved - for example, if arbitrary real displacements of the midpoint are allowed. The usual way the algorithm has been stated has been with a random number generator determining the displacements. In practice, however, only small tables of random numbers have been used to avoid the speed sacrifice involved in using a random number generator millions of times. In fact, as Plate FOUR.FRACTAL shows, only three different numbers suffice for good-looking mountains. If a random number generator is used at all, its only role is to provide a nice spread to the finite set of numbers to be used repeatedly. Plate THREE.THISTLE shows a similar result for graftal plants, the system 0.1.0.1[1].0.11.0.0 in this case.

A small (finite) number of different displacements with only local effect suggests formal language theory again. We can expand the one rule in Figure CARPENTER to a finite set of rules corresponding to one left side for every possible triple of symbols, each of which represents one of the allowed displacement factors. A midpoint displacement becomes a function of its two nearest node labels, so shared edges are forced to behave the same way.

These mountains suffer from a defect known as the "creasing problem". The midpoint displacements are in height only to avoid foldover and self-intersection. Thus the initial database lines are never broken out of recognition by the subdivision process, especially if observed along their initial directions. In formal language terminology, as Loren Carpenter has pointed out to me, the problem with his language is that it is context-free. Information internal to an original database triangle is never passed to neighboring triangles. An open question is whether a context-sensitive grammar exists for mountains which avoids the creasing problem. Loren believes so and has designed a context-sensitive grammar which he is currently testing. Of course, a context-sensitive neighborhood must be finite (cf. local); it is already known that a global approach will work[23], but our goal is to find a computationally more satisfactory approach. As anyone can testify who lives near the edge of a tectonic plate, creases in landscapes are natural. The problem is to bring them under control.

Figure CARPENTER. The production rule for a Carpenter mountain.

Figure KOCH. The production rule for a Koch (or Cesàro) curve.

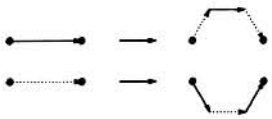

Figure SIERPIŃSKI. The production rules for a Sierpiński arrowhead.

LANGUAGES IN MANDELBROT'S BOOK

Mandelbrot's book is filled with formal language examples informally presented. At first glance they appear to be of a very simple kind. An example is the production rule for the so-called Koch curve shown in Figure KOCH. If the axiom is an equilateral triangle, then the first generation is a Star of David in this language (cf. [14], p. 42). Arrowheads have been added to the line segments in the production rules whenever direction is important.

All of the languages in Mandelbrot use only one symbol in the production rule, a solid line segment, but the production rules are frequently augmented with non-grammatical rules such as "the generator‡ must be made to alternate between the left and the right". These additional remarks can be handled entirely by a formal language if additional symbols are allowed. Figure SIERPIŃSKI shows the two production rules on two symbols replacing the one rule on one symbol plus additional stipulation in [14], p. 142. Directionality is important. The use of two symbols and arrows readily captures the notion of swapping from left to right in successive generations. Of course, in the final interpretation of a word, arrowheads are removed and dotted lines replaced with solid lines.

PARALLEL GRAPH GRAMMARS

There are several approaches we might take for defining the class of languages suggested in the preceding sections. The class chosen must answer to the following observations:

‡ Mandelbrot calls the axiom an "initiator" and the right-hand side of a production rule a "generator".

The context-free plant productions require the form tree-replaces-segment. There must be a way of specifying connectivity during replacement. Several symbols must be allowed. Directionality is important. Preservation of branching is important, but the actual angles taken and the lengths of branch segments are immaterial to the language generation; they matter greatly at the interpretation step, of course. Thus the tree topology is important but the geometrical scaling of parts is not important. Similarly, the production rules are applied to segments of any length and rotation, and hence are topological. The context-sensitive plant productions require the same form but require a mechanism for specifying context of the replaced segment.

The Carpenter mountain language requires a graph-replaces-graph form, with attachment information for before and after replacement. It is context-free, however. The geometry is derived from the finite alphabet of displacement factors in an interpretation postprocessing step. Only the topology of the language is important. Directionality is not important in this case.

Directionality is important for most of the languages in Mandelbrot's book. The replacement rules are of the form graph-replaces-arc, although tree-replaces-arc usually suffices. Geometry is always important in these languages. In analogy with the other languages in this paper, I have used a grammar to generate the data *representation* and assumed a postprocessing step for data *interpretation*. For the languages in Mandelbrot, the interpretation step is mostly cosmetic - removing arrowheads and substituting solid lines for dotted lines. The geometry is carried with the production rules which means that they must work under affine transformation. These grammars might be most appropriately described as *parallel picture grammars* which are parallel graph grammars with an enforced geometry.

It is difficult to find a single formal grammar definition that will capture all of the aspects above, so my intention is to give an intuitive definition for graftals indicating a possible form for the formal definition of the representation. Formalization of the crucial interpretation step must await further work.

There have been several attempts at extending 1-dimensional L-systems to graphs[4,5,15,16,18]. It turns out to be quite difficult to do concisely. The general notion is based on a picture being a bounded subset of E^2 or E^3 that may have color. It is replaced with another picture, perhaps empty, with the same bounds. The rules are translation, scale, and rotation invariant. Thus whenever the left side of a rule is found to apply to a picture - by thinking of it as a template which is moved about over a "word", or another bounded, colored subset of space, by affine transformation - it is replaced by the right side of the rule transformed by the same transformation required for the template match.

The replacement rules are easy to specify in the context-free case. In the context-sensitive case, a neighborhood is defined to be a picture as above, and only a subset of it, its *kernel*, is replaced by a right-side picture. The kernel and the right-side picture must have the same bounds. The entire neighborhood (including the kernel) must match as a template before the kernel is eligible for replacement.

Graph grammars use graphs instead of pictures. Templates are replaced by "coverings" of labeled nodes by labeled nodes and labeled arcs by labeled arcs, so matches are determined on the basis of connectivity and label matches. The main difficulty in defining how the replacement rules work is that of defining the connectivity before and after. We have shown how to solve this problem for very simple graphs in the examples of this paper.

A very general approach, based on the "push-out" construction of category theory, is presented in Ehrig and Kreowski[4] which generalizes the pioneering paper of Ehrig, Pfender, and Schneider[3] to the parallel case. The restriction of the general case to injective mappings, or embeddings, might suffice for our purposes. An even more restrictive approach which works for all examples here except the Carpenter mountains is the "handle substitution parallel graph grammars" of Ehrig and Rozenberg[5], where a "handle" is a graph arc together with its source and target nodes. Two paper collections[1,13] and, in particular, the two surveys by Ehrig[6] and Nagl[17] are good starting points for graftal investigations.

PARTICLE SYSTEM PLANTS

Bill Reeves has written a program for generating trees, the output of which is shown in Plates ASPEN.SPRUCE, MEADOW, and MAXFIELD[21]. These form another class of rich, complex, natural objects which are related to particle systems in their use of thousands of randomly controlled particles, or leaves, and to graftals in their use of small initial descriptions which are used to generate elaborate tree structures of high complexity. A major difference is the use of randomness throughout (but we shall not be surprised if finite sets of well-selected numbers work as well). Each tree is essentially a data structure with many elements, each of which controls one of the random processes used to realize the tree at rendering time. These include height, width, branching angles, bending factors, number of branches, and coloring. For example, an aspen tree is specified with about 120 parameters. Another difference is the lack of an obvious way to obtain plant growth.

So far as similarities are concerned, besides sharing the database amplification property with graftals, these trees have the "closer you get, the more you see" property in common with them due to the recursive leaf generation feature which causes more leaves to be rendered as the screen space occupied by the tree gets larger.

The lighting model, derived in conjunction with Tom Duff and Rob Cook, approximates the extensive self-shadowing of plants by darkening from the surface toward the trunk, modulated by branch density. This is in addition to the surface lighting obtained by assuming the outer envelope of the tree is shaded by conventional models. Neighboring trees also cast shadows determined from their envelopes - e.g., ellipsoids or cones. These lighting techniques go through for the graftals if an approximating envelope can be extracted.

SUMMARY

A rich class of plant models is presented here. The models gain their interest from great visual complexity approaching that of Nature's plant kingdom. The complexity is gained with little human effort by the exploitation of the database amplification property of formal languages, generalized from strings to graphs. These plants can be represented with efficient data structures, and they can be made to grow and flower in time.

Failure of the plants to qualify as fractals, plus the fact that they share much of the spatial complexity of fractals, led to the definition - intuitive, so far - of the graftals, a family of objects generated by parallel graph grammars and including many of the well-known fractals. They do not, however, have to be fractal - i.e., have Hausdorf-Bescovitch dimension greater than topological dimension. Because they are, in general, not strictly self-similar, the Hausdorf-Bescovitch dimension is difficult to compute. It is suggested that fractality itself is not the important property of fractals for Computer Imagery but that it is their well-controlled database amplification property that is of such great usefulness. This is a feature of the graftal family in general, the amplification being controlled by a finite - sometimes surprisingly small - set of grammar rules.

The particle system plants of Bill Reeves were introduced to emphasize that neither graftals nor fractals cover the gamut of natural form generation systems. What all these systems have in common is the database amplification property which is very important for the construction of satisfyingly complex scenes in reasonably short times. They all also share a loose notion of "space filling" with "self-similar" constructions but not strictly enough (in the nonfractal cases) to jump into higher dimensionality.

It has been demonstrated that randomness is not necessary for pictures of interesting complexity. Randomness is just a convenient way of generating a well-dispersed set of numbers, instead of doing it by hand. Tom Duff has pointed out that the pseudo-random number generators used in computer science are based on a related principle: From a small set of generating rules, they supply strings of sufficient length and complexity to simulate true randomness. There are measures of how close to random these finite sequences are. This raises the question of what is sufficient complexity to satisfy perceptual and esthetic requirements of human beings.

It is well-known that a set recognized by a Turing machine is equivalent to a language generated by a type **0** grammar. Thus the operation of a computing machine is strongly related to the generation of language by a formal grammar. The formal language approach advocated here renders the computational processes of such a machine as objects - the state of a computation is frozen and called a picture. Strictly speaking, it is a data representation of a picture. The representation and the thing represented are kept separate. After the machine has done its work, the artist may step in and modulate the computed form with esthetic judgment, thus becoming the composer of the image.

ACKNOWLEDGEMENTS

My colleagues in the Graphics Project have aided me with their usual zest. Specifically, Loren Carpenter and Bill Reeves contributed important comments and pictures to the work, Rodney Stock and David Salesin discussed the creasing problem with me at length, and Rob Cook and Tom Porter are authors of the principal rendering software used. Ed Catmull, Tom Duff, and Andrea Kaufman were helpful critics as well.

REFERENCES

1. Claus, Volker, Hartmut Ehrig, and Grzegorz Rozenberg (Editors), *Lecture Notes in Computer Science No. 73: Graph-Grammars and Their Application to Computer Science and Biology,* Springer-Verlag, Berlin/Heidelberg/New York (1979). Proceedings of the conference held at Bad Honnef, West Germany, October 30-November 3, 1978.
2. Cook, Rob, Loren Carpenter, Thomas Porter, William Reeves, David Salesin, and Alvy Ray Smith, *Road to Point Reyes*, By the Lucasfilm Computer Graphics Project. Title page picture for SIGGRAPH '83 Proceedings. July 1983.
3. Ehrig, Harmut, M. Pfender, and H. J. Schneider, "Graph-Grammars: An Algebraic Approach," pp. 167-180 in *Proceedings of 14th Annual Symposium on Switching & Automata Theory* (October 1973). Now known as the Symposium on the Foundations of Computer Science.
4. Ehrig, Harmut and H.-J. Kreowski, "Parallel Graph Grammars," pp. 425-442 in *Automata, Languages, Development*, ed. Aristid Lindenmayer and Grzegorz Rozenberg, North-Holland Publishing Company, Amsterdam/New York/Oxford (1976).
5. Ehrig, Harmut and Grzegorz Rozenberg, "Some Definitional Suggestions for Parallel Graph Grammars," pp. 443-468 in *Automata, Languages, Development*, ed. Aristid Lindenmayer and Grzegorz Rozenberg, North-Holland Publishing Company, Amsterdam/New York/Oxford (1976).
6. Ehrig, Hartmut, "Introduction to the Algebraic Theory of Graph Grammars (A Survey)," pp. 1-69 in *Lecture Notes in Computer Science No. 73: Graph-Grammars and Their Application to Computer Science and Biology*, ed. Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg, Springer-Verlag, Berlin/Heidelberg/New York (1979).
7. Fournier, Alain, Don Fussel, and Loren C. Carpenter, "Computer Rendering of Stochastic Models," *Communications of the ACM* **25**(6), pp. 371-384 (June 1982).
8. Herman, Gabor T. and Grzegorz Rozenberg, *Developmental Systems and Languages,* North-Holland Publishing Company, Amsterdam/New York/Oxford (1975).
9. Hogeweg, Pauline and B. Hesper, "A Model Study on Biomorphological Description," *Pattern Recognition* **6**, pp. 165-179, Pergamon Press (1974).

10. Hopcroft, John E. and Jeffrey D. Ullman, *Formal Languages and Their Relation to Automata,* Addison-Wesley Publishing Company, Menlo Park, California (1969). The latest edition of this book is entitled *Introduction to Automata Theory, Languages, and Computation*, 1979, and includes a small section on L-systems.
11. Kawaguchi, Yoichiro, "A Morphological Study of the Form of Nature," *Computer Graphics* **16**(3), pp. 223-232 (July 1982). SIGGRAPH '82 Proceedings.
12. Lindenmayer, Aristid, "Mathematical Models for Cellular Interactions in Development, Parts I and II," *Journal of Theoretical Biology* **18**, pp. 280-315 (1968).
13. Lindenmayer, Aristid and Grzegorz Rozenberg (Editors), *Automata, Languages, Development,* North-Holland Publishing Company, Amsterdam/New York/Oxford (1976). Proceedings of the conference held at Noordwijkerhout, The Netherlands, March 31-April 6, 1975.
14. Mandelbrot, Benoit, *The Fractal Geometry of Nature,* W. H. Freeman and Company, San Francisco (1983). The 1983 printing differs from the 1982 printing by the addition of a small section at the end.
15. Mayoh, Brian H., "Another Model for the Development of Multidimensional Organisms," pp. 469-485 in *Automata, Languages, Development*, ed. Aristid Lindenmayer and Grzegorz Rozenberg, North-Holland Publishing Company, Amsterdam/New York/Oxford (1976).
16. Nagl, Manfred, "On a Generalization of Lindenmayer-Systems to Labelled Graphs," pp. 487-508 in *Automata, Languages, Development*, ed. Aristid Lindenmayer and Grzegorz Rozenberg, North-Holland Publishing Company, Amsterdam/New York/Oxford (1976).
17. Nagl, Manfred, "A Tutorial and Bibliographical Survey on Graph Grammars," pp. 70-126 in *Lecture Notes in Computer Science No. 73: Graph-Grammars and Their Application to Computer Science and Biology,* ed. Volker Claus, Hartmut Ehrig, and Grzegorz Rozenberg, Springer-Verlag, Berlin/Heidelberg/New York (1979).
18. Paz, Azaria, "Multidimensional Parallel Rewriting Systems," pp. 509-515 in *Automata, Languages, Development*, ed. Aristid Lindenmayer and Grzegorz Rozenberg, North-Holland Publishing Company, Amsterdam/New York/Oxford (1976).
19. Reeves, William T., "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects," *ACM Transactions on Graphics* **2**(2), pp. 91-108 (April 1983).
20. Smith, Alvy Ray, Loren Carpenter, Pat Cole, Tom Duff, Chris Evans, Thomas Porter, and William Reeves, "Genesis Demo," in *Star Trek II: The Wrath of Khan*, Paramount (June 1982). Created by the Lucasfilm Computer Graphics Project for Industrial Light and Magic.
21. Smith, Alvy Ray, Loren Carpenter, Ed Catmull, Rob Cook, Tom Duff, Craig Good, John Lasseter, Eben Ostby, William Reeves, and David Salesin, *Andre' and Wally B.*, Created by the Lucasfilm Computer Graphics Project. July 1984.
22. Stevens, Peter S., *Patterns in Nature,* Little, Brown and Company, Boston (1974).
23. Voss, Richard F., *Fractal Lunar Mist*, Cover picture for SIGGRAPH '83 Proceedings. July 1983.

Plate ASPEN.SPRUCE. Aspen and spruce trees by Bill Reeves. These are related to particle systems and graftals.

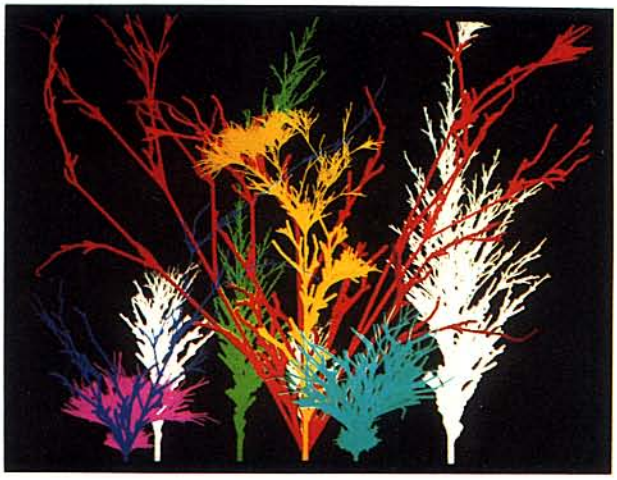

Plate BUSHES. Several context-sensitive species without flowers or leaves.

Plate CARTOON.TREE. A 2-D rendering of the context-free grammar in Figure FREETREE.

Plate GARDEN. Several context-sensitive graftal species showing the variety easily obtained.

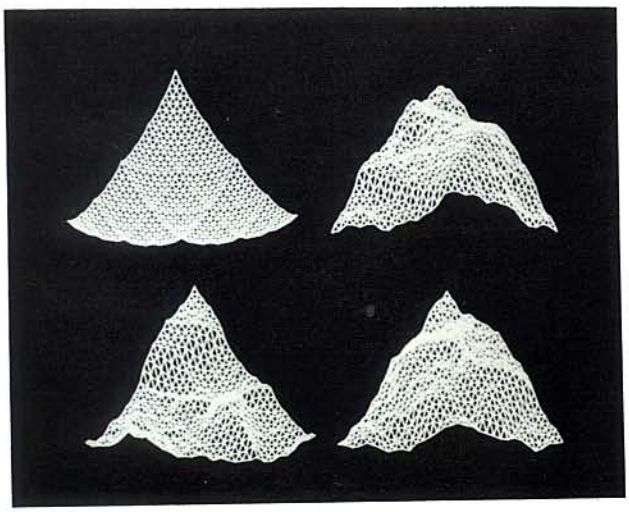

Plate FOUR.FRACTAL. The upper left mountain results from only one subdivision factor. It is unsatisfactory. The upper right mountain resulted from three carefully chosen factors, and the lower left from five. They both work. The lower right mountain is generated using random factors. It is clear that deterministic mountains (e.g., the upper right one) suffice.

Plate GARDEN.DROP. Several more examples of flowering species, with dropped shadows.

Plate GREEN.FLAME. Several 2-D renderings of the grammar in Figure SENSTREE, varying the parameters of the rendering program **Gene**.

Plate MEADOW. More computer generated aspen and spruce trees, seen against a meadow handpainted by John Lasseter.

Plate MAXFIELD. Computer generated aspen and spruce trees, with coloring inspired by Maxfield Parrish, in a fern-covered glen painted by John Lasseter with a computer painting program.

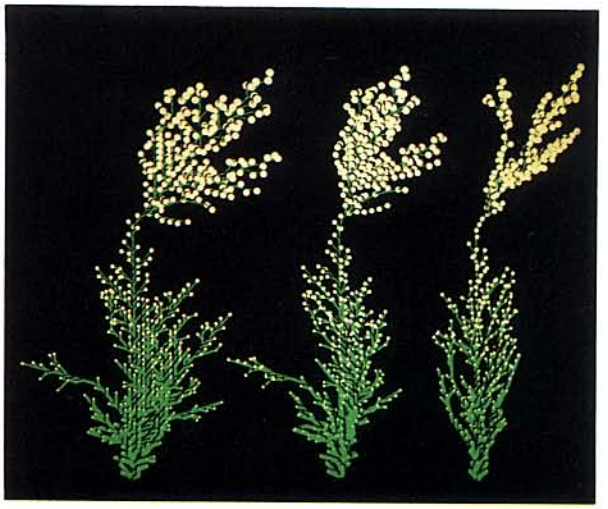

Plate THREE.THISTLE. The plant on the left has only one elevation angle (actually two, but one is the negative of the other); this plant is unsatisfactorily regular. It has only one azimuth angle also. The center plant has only two elevation angles and two azimuth angles. It works. The plant on the right is fully random. Every angle (azimuth and elevation) is distinct. It works too. But the principal result illustrated is that fully deterministic plants (e.g., the center one) suffice.

Plate VITA.PLANTS. Several variations obtained with the **Gene** program.

Plate WISP.GROWTH. Frames 20, 40, 60, 80, and 120 from a growing plant movie. The plant is also slowing rotating.

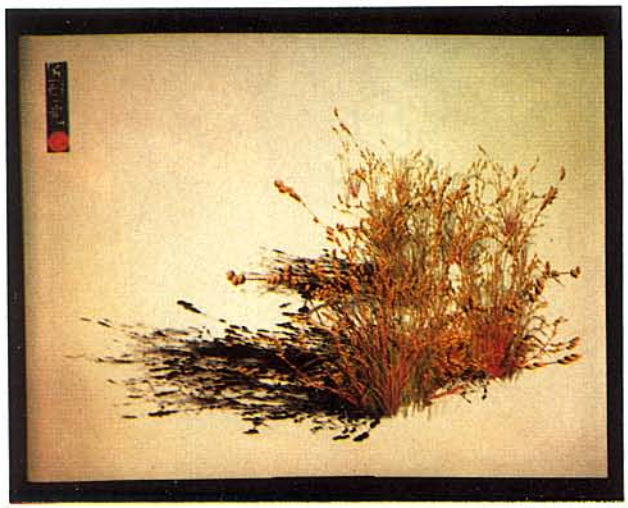

Plate WHITE.SANDS. Several 3-D renderings of the context-sensitive grammar 0.0.0.11.1.1[1].1.0 mixed with particle system grasses.

Plate WITH.WITHOUT. A 2-D rendering of the grammar in Figure SENSTREE, showing the 35th generation with and without flowers.