Real Time Design and Animation of Fractal Plants and Trees

Peter E. Oppenheimer
New York Institute of Technology
Computer Graphics Lab

ABSTRACT

The goal of science is to understand why things are the way they are. By emulating the logic of nature, computer simulation programs capture the essence of natural objects, thereby serving as a tool of science. When these programs express this essence visually, they serve as an instrument of art as well.

This paper presents a fractal computer model of branching objects. This program generates pictures of simple orderly plants, complex gnarled trees, leaves, vein systems, as well as inorganic structures such as river deltas, snowflakes, etc. The geometry and topology of the model are controlled by numerical parameters which are analogous to the organism's DNA. By manipulating the *genetic* parameters, one can modify the geometry of the object in real time, using *tree based* graphics hardware. The random effects of the environment are taken into account, to produce greater diversity and realism. Increasing the number of significant parameters yields more complex and evolved species.

The program provides a study in the structure of branching objects that is both scientific and artistic. The results suggest that organisms and computers deal with complexity in similar ways.

CR CATAGORIES AND SUBJECT DESCRIPTORS: C.3 [Special-Purpose and Application-Based Systems]: Real-time systems; G.3 [Probability and Statistics]: Random number generation; I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.6.0 [Computer Graphics]: Simulation and Modeling; J.3 [Life and Medical Sciences]: Biology; J.5 [Arts and Humanities]: Arts, fine and performing.

ADDITIONAL KEY WORDS AND PHRASES: plant, tree, fractal, real time, DNA, genetics animation, recursion, stochasitic modeling, natural phenomena

1. INTRODUCTION

Benoit Mandelbrot recognized that the relationship between large scale structure and small scale detail is an important aspect of natural phenomena. He gave the name *fractals* to objects that exhibit increasing detail as one zooms in closer. [12][13] If the small scale detail resembles the large scale detail, the object is said to be self-similar.

The geometric notion of fractal self-similarity has become a paradigm for structure in the natural world. Nowhere is this principle more evident than in the world of botany. Recursive branching at many levels of scale, is a primary mechanism of growth in most plants. Analogously, recursive branching algorithms, are fundamental to computers. Many high performance processing engines specialize in tree data structures.

Computer generation of trees has been of interest for several years now. Examples of computer generated trees include

Benoit Mandelbrot (1977)(1982) [12],[13]
Marshall,Wilson,Carlson (1980) [14]
Craig Reynolds, (1981) [19]
Yoichiro Kawaguchi (1982) [10]
Geoff Gardiner (1984) [9]
Aono,Kunii (1984) [2]
Alvy Ray Smith, Bill Reeves (1984)(1985) [21][18]
Jules Bloomenthal (1985) [4]
Demko,Hodges,Naylor (1985) [6]
Eyrolles, Francon, Viennot (1986) [8]
among others.

Not all of the above used recursive tree data structures. Mandelbrot is largely responsible for the growing awareness of recursion as a process in nature. He and Kawaguchi each used minimal recursive topologies and added simple geometric relationships to generate complex images of branching plants and other objects. Alvy Ray Smith applied the theory of formal languages to generate more complex and systematized tree topologies, while emphasizing the separation between topology and geometry.

Each attempt at modeling trees takes a new approach, and thereby captures some different aspect of branching phenomena.

2. THE TREE MODEL

The tree model presented in this paper has the following features:

- A detailed parameterization of the geometric relationship between tree nodes.
- Real Time Design and Animation of tree images using high performance hardware.
- Application of Stochastic (random) Modeling to both topological and geometric parameters.
- Stochastic modeling of tree bark.
- High Resolution (2024 x 1980) Shaded 3D Renderings.

Here's how the model works:

This program implements a recursive tree model. Each tree generated satisfies the following recursive tree node definition:

```
tree :=
{
        Draw Branch Segment

        if (too small)
                Draw leaf
        else
        {
                # Continue to Branch
                {
                        Transform Stem
                        "tree"
                }
                repeat n times
                {
                        Transform Branch
                        "tree"
                }
        }
}
```

Paraphrased, a tree node is a branch with one or more tree nodes attached, transformed by a 3x3 linear transformation. Once the branches become small enough, the branching stops and a *leaf* is drawn. The trees are differentiated by the geometry of the transformations relating the node to the branches and the topology of the number of branches coming out of each node. These branching attributes are controllable by a set of numerical parameters. Editing these parameters, changes the tree's appearance. These parameters include:

- The angle between the main stem and the branches
- The size ratio of the main stem to the branches
- The rate at which the stem tapers
- The amount of helical twist in the branches
- The number of branches per stem segment

Figure 1 shows a simple example.

3. RANDOM NUMBERS IN FRACTAL MODELING

If the parameters remain constant throughout the tree, one gets a very regular looking tree such as a fern. This tree is strictly self similar; that is, the small nodes of the tree are identical to the top level largest node of the tree.

If the parameters vary throughout the tree, one gets an irregular gnarled tree such as a juniper tree. In order to achieve this, each parameter is given a mean value and a standard deviation. At each node of the tree, the parameter value is regenerated by taking the mean value and adding a random perturbation, scaled by the standard deviation. The greater the standard deviation, the more random, irregular, and gnarled the tree. The resulting tree is *statistically* self-similar; not *strictly* self-similar.

There are several reasons for the stochastic approach. First, adding randomness to the model generates a more natural looking image. Large trees have an intrinsic irregularity (caused in part by turbulent environmental effects). Random perturbations in the model reflect this irregularity. Second, random perturbations reflect the diversity in nature. A single set of tree model parameters can generate a whole forest of trees, each slightly different. This increased database amplification is one of the hallmark features of fractal techniques.

When generating random numbers, one must be careful about consistency and reproducibility. Simply reseeding the random number generator at the outset is not enough. For example, if one decreases the *minimum branch size* parameter, one would like the new tree to be a simple extension of the old tree. However, the order in which these new branch nodes are generated is intermixed with the old branch nodes. This could potentially scramble the random number generator, drastically changing the shape of the resulting tree. To avoid this, the random perturbations corresponding to a given node, should be determined by the *topological branch address* of the node, and not the order in which it is generated.

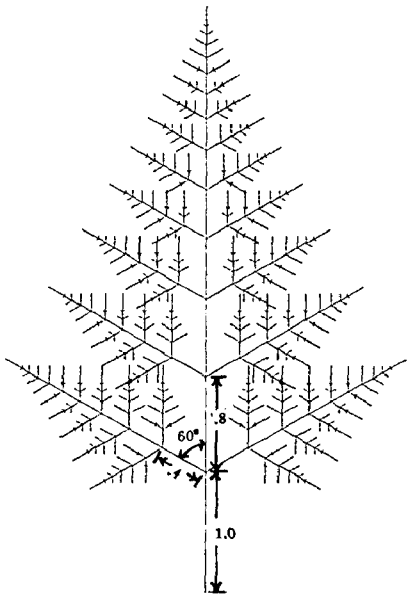


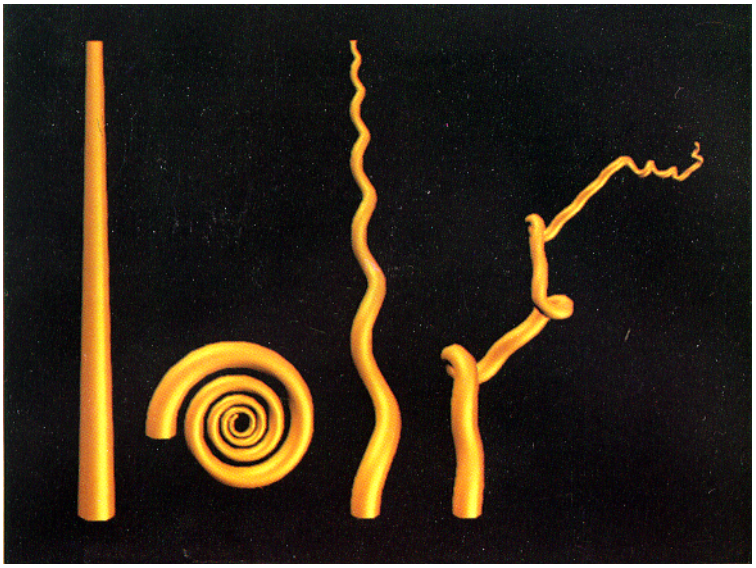
Figure 1

stem/stem ratio = .8
branch/stem ratio = .4
branching angle = 60°

Fallen Leaf - An exaggerated image of vein branching in a fall leaf. The external boundary shape is the limit of growth of the internal veins. (512x480)

Snowflakes - Self similar branching occurs in the inorganic world as well.(512x480 resolution)

Fractal Fern - The classic organic self similar form. Every branch point looks the same -- but at different scales. (1024 x 960)



(a) (b) (c) (d)

Figure 2 Spirals & Helixes

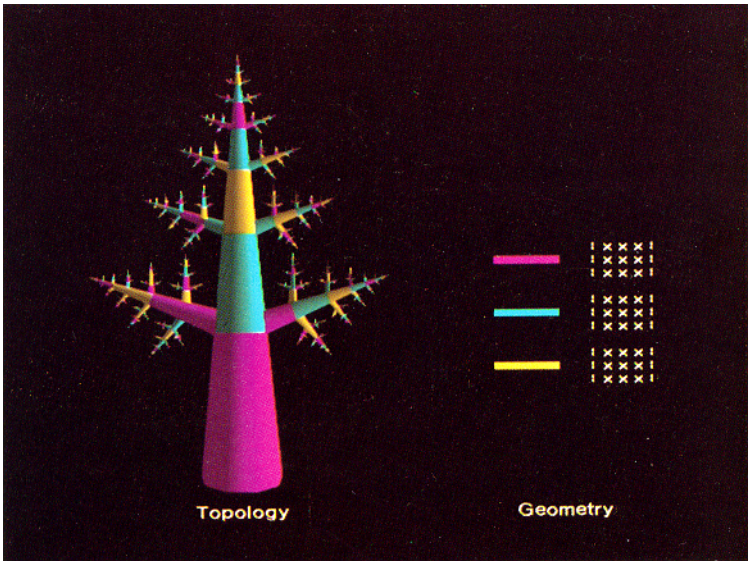


Figure 3 The Display List

Decreasing the *minimum branch size* parameter can also cause branches to *POP* on the tree, causing a disturbing effect in animations. This can be avoided by drawing partial branch segments whenever scaling a branch segment would cause it to be smaller than the *minimum branch size*. Using this method causes the branches to grow smoothly as the minimum branch size parameter is decreased, rather than *popping on*.

4. MODELING STEM SHAPE

By stripping all of the branches one is left with just a stem. By varying the transformation between stem segments, one derives the class of *spirals and helizes* and their random perturbations. These shapes appear in all forms of growth, organic and inorganic -- from the inner ear, sea shells, and plant sprouts, to spiral galaxies. *Spirals and helizes* are in some sense degenerate self similar sets. They are the atomic units that make up the fractal trees.

Figure 2 shows 4 typical *stem* shapes.

a) *cylinder*: the transformation is a translation and a scale.

b) *spiral*: one performs a rotation perpendicular to the stem axis, in addition to a scale and translation.

c) *heliz*: one performs an additional rotation along the stem axis.

d) *squiggle* By randomly changing the transformation from segment to segment, case c) becomes case d).

Branches are simply stem shapes attached to the main stem and each other.

5. RENDERING THE FRACTAL

A variety of geometric elements can be used to render the branch segments. The simplest primitive is one single vector line per tree node. Antialiased vector lines with variable thickness allow one to taper the branches towards the tip. This method is satisfactory for leaves, ferns and other simple plants, for small scale detail in complex scenes, or for more abstract stylized images. Varying the vector color provides depth cuing and shading, and can also be used to render blossoms or foliage. Antialiased vectors are similar to particle systems used by Bill Reeves in his forest images. [18]

The thicker branches of a tree require a shaded 3D primative. Bump mapped polygonal prisms are used to flesh out the trees in 3D. The program makes sure that the polygons join continuously along each limb. The branches emanating from a limb simply interpenetrate the limb. For a more curvilinear limb shape, one can link several prisms together between branch points.

Shaded polygon limbs are far more computationally expensive than antialiased vectors. Since the number of branches increases exponentially with branching depth, one can spend most of an eternity rendering sub pixel limb tips, where bark texture and shading aren't visible anyway. In addition to being faster, sub pixel vectors are easier to antialias than polygons on our available rendering package. So for complex trees with a high level of branching detail, polygonal tubes were used for the large scale details, changing over to vectors for the small stuff. One can notice the artifacts of this technique. Overall, however, the eye ignores this inconsistency if the cutover level is deep enough. Thinner branches require fewer polygons around the circumference; in fact triangular tubes will do for the smallest branches.

6. BARK

Sawtooth waves modulated by Brownian fractal noise are the source for the simulated bark texture. This texture is bump mapped onto the tree limbs. More specifically,

$$\text{bark}(x,y) = \text{saw}\,[\,N * [x + R * \text{noise}(x,y)]]$$

where

$$\text{saw}(t) = \begin{cases} 2 * \text{fraction}(t) & \text{if fraction}(t) < .5 \\ 2 * (1 - \text{fraction}(t)) & \text{if fraction}(t) > .5 \end{cases}$$

noise (x,y) is a periodic (ie wrapping) fractal noise function.

N is the number of bark ridges, and
R is the roughness of the bark.

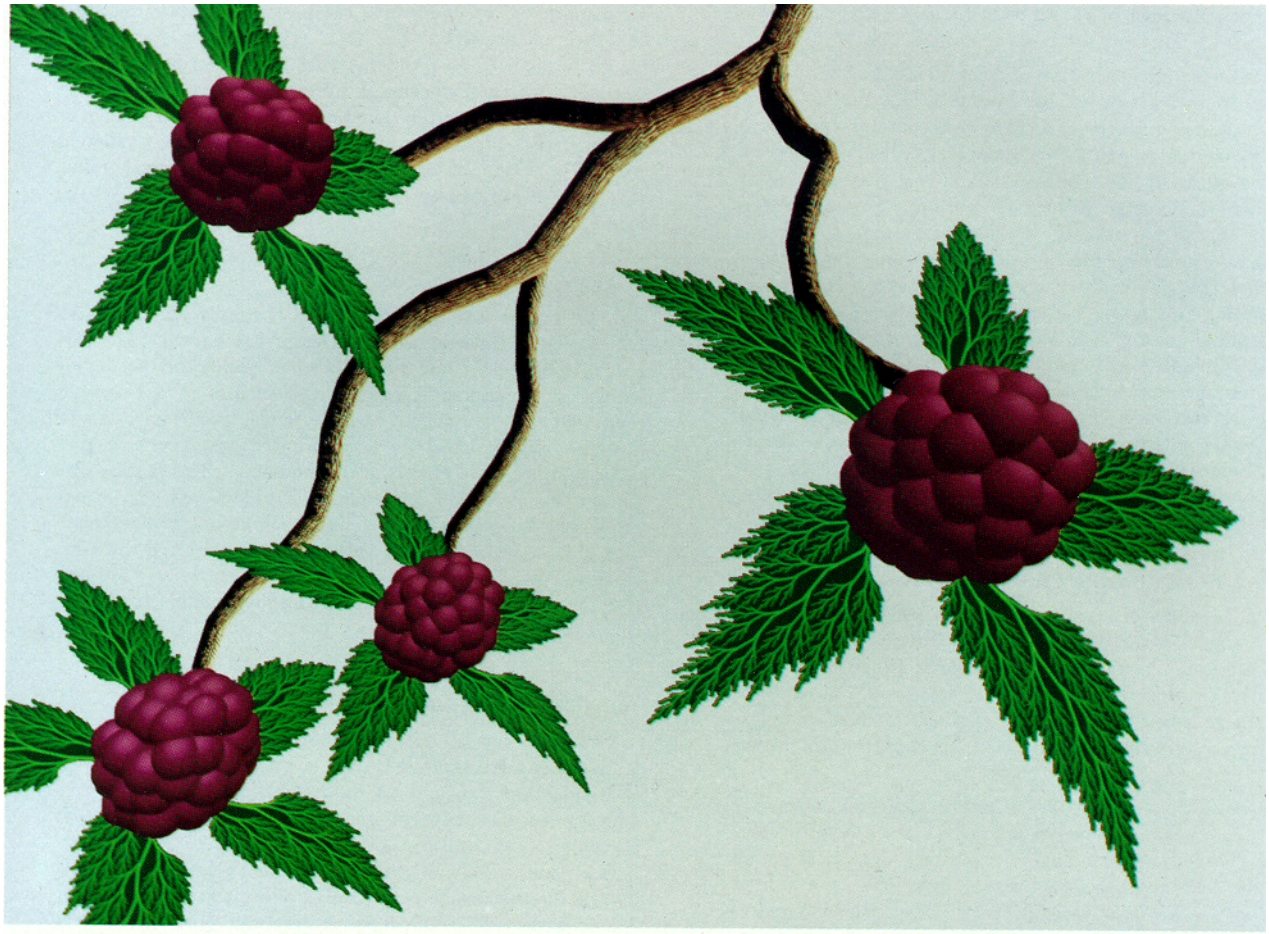
Paraphrased, bark is generated by adding fractal noise to a ramp, then passing the result through a sawtooth function. A close up view of the bark would look like the ridges of a fractal mountain range. By adding the noise before the sawtooth function, the crests of the sawtooth ridges become wiggly. Note that bark(x,y) wraps in x and y.

7. REAL TIME FRACTAL GENERATION

Complex tree images can take 2 hours or more to render on a VAX 780. Editing tree parameters at this rate is not very effective. Near real time feedback is needed to allow one to freely explore the parameter space, and design the desired tree. Since vertex transformation cost is high for a complex fractal tree, hardware optimized for linear transformations was used for the real time editor. The Evans and Sutherland MultiPictureSystem generates vector drawings of complex 3D display lists in near real time. The display lists on the MPS look a lot like our tree nodes: primitive elements, transformed by linear transformations, and linked by pointers to other nodes. For a strictly self similar tree, the transformation is constant, therefore the entire display list can share a single matrix. To modify the tree one only has to change this one matrix, rather than an entire display list. This makes updating the tree display list very fast. For non-strictly self similar trees, the transformations are not the same. However a lookup table of less than a dozen transformations, is adequate to provide the necessary *pseudo* randomness. [21]

Figure 3 illustrates the logic of the display list.

The left side of the display list contains the topological description of the tree. Each node contains pointers to offspring nodes plus a pointer to the transformation matrix which relates that node to its parent node. This part of the display list is purely topological: it contains no geometric data. The geometric information is contained in the small list of transformation matrices on the right. To edit a tree, one can create a large topology list once and then rapidly manipulate only the small geometry list. Alvy Ray Smith in his research on *graftals*, recognized the separation of the topological and geometric aspects of trees. He calls these components the graph, and the interpretation respectively. His work deals primarily with specification of the tree topology, ignoring interpretation for the most part [21]. The



Views I - By randomly perturbing the branching parameters, one generates a more naturalistic gnarled tree. (2048x1920)

Blossomtime - The bark of this cherry tree is made with bump mapped polygonal tubes. The blossoms are colored vectors (particle systems) Instead of modeling blossoms, one can simply *dip* the branches in pink paint. (1024x960)

Raspberry Garden at Kyoto - The leaves and branches are generated by the fractal branching program. Berries are based on symmetry models by Haresh Lalvani, with added random perturbations. Non-self similar features such as berries, require genetic specialization. (1024x960)

paper presented here emphasizes the geometric interpretation. The thesis of this paper is that the key to realistically modeling the diversity of trees lies in controlling the geometric interpretation. Many different topologies were used in this project. But by varying the geometric interpretation of a *single* topology, one could still generate a wide variety of trees each with its own distinct taxonomic identity.

The real time generation of fractal trees has been packaged as an interactive editing system. This multi-window system allows one to edit the tree parameters, (both geometric and topological) via graphically displayed sliders. A vector image of the tree responds in real time. To see all the parameters change at once one performs keyframe interpolation of the parameters. Each tree parameterization is written to a keyframe file. A cubic spline program, interpolates these parameters to create the inbetween frames. In the resulting animation, the tree metamorphoses from key to key. A simple *tree growth* animation is achieved by interpolating the trunk width parameter, and the recursion size cutoff parameter. Modifying additional parameters makes the growth more complex and natural looking. For example, many plants uncurl as they grow. A metamorphosis animation is achieved by interpolating parameters from different tree species. Growing and metamorphosing tree animations appear in *The Palladium* animation produced at NYIT. [1]

8. FRACTALS, COMPUTERS, AND DNA

The economic advantage of this program is that a highly complex structure is generated from a simple concise kernel of data which is easy to produce. (Such large database amplification is a primary advantage of fractal techniques in general.) How does the representation of complexity by computers compare with complex expression in nature itself?

Presumably, complexity in nature has evolved because it can bestow benefits on an organism. But, as with computers, complexity must not be a burden. Genetic economy demands that intricate structures be described by a limited supply of DNA. This struggle to simplify genetic requirements, determines the geometric structure of the plant. Form follows genetic economics.

This suggests a natural explanation as to why self-similarity abounds in the natural world: evolution has resolved the tension between complexity and simplicity in the same way that computer science has -- with recursive fractal algorithms. If fractal techniques help the computer resolve the demands of database amplification, then presumably organisms can benefit as well. For genes and computers alike, self-similarity is the key to thrifty use of data. [16][13]

The parameters of the tree program are numerical counterparts to the DNA code that describes a tree's branching characteristics. The logic of the gene is mimicked although the mechanism is different. The early stages of the model contained only 3 changeable parameters. The resulting images were of very simple fern like plants. New species were generated by controlling the parameter values, rerolling the dice of mutation and then selecting the forms that would be allowed to proliferate. As the model became more complex with the inclusion of more parameters, the program created images of more genetically complex trees such as cherry trees, higher on the evolutionary scale. Whereas natural selection of organisms is based on survival value, this aesthetic selection is based upon resemblance to the forms of nature.

9. CONCLUSION

Of course any scientific model is simply an attempted translation of nature into some quantifiable form. The success of the model is measured by some qualifyably predictive result. In experimental science, the success of a theory is measured by the degree to which the predicative model matches experimental data. Computer graphics now provides another style of predictive modeling. The success of a computer simulation is reflected in how well the image resembles the object being modeled. If one can model a complex object through simple rules, one has mastered the complexity. What appeared to be complex, proves to be primitive in the end. And the proof (although subjective) is in the picture.

10. ACKNOWLEDGEMENTS

Thanks go to all the members of NYIT staff and management for their help in the production of this work. Particular mention goes to Paul Heckbert and Lance Williams for software support and consultation. Kevin Hunter's Brownian Fractal Noise Texture was used to generate the bark. Haresh Lalvani contributed the original symmetry models for the raspberries. Design and direction by Rebecca Allen made the *Palladium* Animation possible. Special thanks go to Jane Nisselson and Robert Wright for editing, and to Ariel Shaw, Cydney Gordon, and the NYIT photo department, for film support.

Special thanks go to Benoit Mandelbrot, who has been a tremendous inspiration and influence.

Winter - Snow Algorithm by Lance Williams. One renders the tree twice. Once with a side light source. Once with an overhead *snow* source. The snow *sticks* to the bump-mapped texture.

Views II - (2048 x 1920)

REFERENCES

[1] Allen, R., Oppenheimer, P., *The Palladium* (Video), New York Institute of Technology, 1985

[2] Aono, M., Kunii, T.L., *Botanical Tree Image Generation*, IEEE Computer Graphics and Applications, Vol. 4, No. 5, May 1984

[3] Bentley, W.A., Humphreys, W.J., *Snow Crystals*, Dover Publications Inc.,New York, 1962 (Originally McGraw Hill, 1931)

[4] Bloomenthal, J., *Modeling the Mighty Maple*, Computer Graphics, Vol. 19, No. 3, July 1985.

[5] Bloomenthal, J., *Nature at New York Tech*, IEEE Computer Graphics and Applications, Vol. 6, No. 5, May 1986

[5] Cole, V.C., *The Artistic Anatomy of Trees*, Dover Publications Inc., New York, 1965 (Originally Seeley Service & Co, London, 1915)

[6] Demko, S., Hodges, L., Naylor, B., *Construction of Fractal Objects with Iterated Function Systems*, Computer Graphics, Vol. 19, No. 3, July 1985.

[7] De Reffye, P., Edelin, C., Francon, J., Puech, C., *L'Atelier de Modelisation de L'Architecture des Plantes*, (illustrated manuscript),1986

[8] Eyrolles, G., Francon, J., Viennot, G., *Combinatoire pour la Synthese d'Images de Plantes*, Cesta: Deuxieme Colloque Image, Vol. 2, Nice, April 1986

[9] Gardiner, G., *Simulation of Natural Scenes Using Textured Quadric Surfaces*, Computer Graphics, Vol. 18, No. 3, July 1984.

[10] Kawaguchi, Y., *A Morphological Study of the Form of Nature*, Computer Graphics, Vol. 16, No. 3, July 1982.

[11] Klee, P.,*On Modern Art*, (tr. Paul Findlay) Faber Ltd., London, 1948

[12] Mandelbrot, B., *Fractals: Form, Chance and Dimension*, W.H. Freeman and Co., San Francisco, 1977.

[13] Mandelbrot, B., *The Fractal Geometry of Nature*, W.H. Freeman and Co., San Francisco, 1982.

[14] Marshall, R., Wilson, R., Carlson, W., *Procedural Models for Generating Three-Dimensional Terrain*, Computer Graphics, Vol. 14, No. 3, July 1980.

[15] Oppenheimer, P., *Constructing an Atlas of Self Similar Sets* (thesis) Princeton University, 1979.

[16] Oppenheimer, P., *The Genesis Algorithm*, The Sciences, Vol 25, No 5., 1985.

[17] Queau, P., *Eloge de la Simulation*, Champ Vallon, France, 1986

[18] Reeves, W., *Particle Systems--A Technique for Modeling a Class of Fuzzy Objects*, Computer Graphics, Vol. 17, No. 3, July 1983.

[19] Reynolds, C., *Arch Fractal*, Computer Graphics (Front Cover), Vol. 15, No. 3, August 1981.

[20] Serafini, L., *Codex Seraphinianus*, Abbeville, New York, 1983.

[21] Smith, A.R., *Plants, Fractals, and Formal Languages*, Computer Graphics, Vol. 18, No. 3, July 1984.

[22] Stevens, P.S., *Patterns in Nature*, Little, Brown, and Co. Boston, 1974.