Construction of fractal objects with iterated function systems

Article *in* ACM SIGGRAPH Computer Graphics · July 1985

DOI: 10.1145/325165.325245

3 authors, including:

Bruce F. Naylor
University of Texas at Austin
29 PUBLICATIONS **1,633** CITATIONS

SEE PROFILE

Construction of Fractal Objects with Iterated Function Systems

Stephen Demko
School of Mathematics

Laurie Hodges and Bruce Naylor
School of Information and Computer Science

Georgia Institute of Technology
Atlanta, Georgia 30332

Abstract

In computer graphics, geometric modeling of complex objects is a difficult process. An important class of complex objects arise from natural phenomena: trees, plants, clouds, mountains, etc. Researchers are at present investigating a variety of techniques for extending modeling capabilities to include these as well as other classes. One mathematical concept that appears to have significant potential for this is fractals. Much interest currently exists in the general scientific community in using fractals as a model of complex natural phenomena. However, only a few methods for generating fractal sets are known. We have been involved in the development of a new approach to computing fractals. Any set of linear maps (affine transformations) and an associated set of probabilities determines an Iterated Function System *(IFS). Each IFS has a unique "attractor" which is typically a fractal set (object). Specification of only a few maps can produce very complicated objects. Design of fractal objects is made relatively simple and intuitive by the discovery of an important mathematical property relating the fractal sets to the IFS. The method also provides the possibility of solving the inverse problem. given the geometry of an object, determine an IFS that will (approximately) generate that geometry. This paper presents the application of the theory of IFS to geometric modeling.*

1. Introduction

Computer graphics is in the midst of discovering ways to construct computer models of "things". This modeling may be representational in nature, such as modeling logic diagrams; or it may be of physical objects, real or imagined, e.g. buildings, plants, Klein bottles, etc. Within the physical arena, a dichotomy can be made between man-made objects and natural objects. Modeling man-made objects, although not trivial by any means, has benefited from the simplicity of the objects and a reliance upon the well developed mathematics of polynomials. Handling natural or complex objects has been much more difficult.

Currently a broad sector of the scientific community is investigating the modeling of certain natural phenomena by fractal sets. This has been prompted by the successful modeling of a variety of such phenomena as evidenced in [Mand83]. Fractals manifest a high degree of visual complexity. Many natural objects exhibit the property that as one views the object at greater magnifications, more and more structure is revealed. Fractal sets enjoy exactly this property; there is no limit to their structure.

Descriptions of fractals can be found in many sources (e.g. [Mand83]). Fractals are sets whose Hausdorff-Besicovitch dimension, which in general is a real number, differs from their topological dimension. An important class of fractals appear to have the desirable characteristic that only a little information is needed for their specification, and a small increase in the specification increases the visual complexity significantly. Automated generation of the fractal from the specification is often straightforward and fast. While this could also be said of sets defined by polynomials, the difference lies in that

polynomials generate smooth geometry while fractals generally possess infinitely non-smooth, highly structured geometries. To model the same sets by traditional polynomial-based methods would require considerably more specification information.

Despite the importance of fractals, only a few methods are known for their computation. We present here the application to geometric modeling of a new method of specifying and computing fractal sets based on *Iterated Function Systems* (IFS) [Barn84a], [Barn84b], [BEHL84] and [Hutc81]. The method provides effective modeling of certain natural objects (with unlimited detail), as well as a large class of other complex objects. It also provides a very compact representation, efficient computation, and a very small amount of user specification. In addition, IFS's have the advantage of a single specification method to obtain a very large class of fractals, where "class" refers to a significant difference in subjective visual properties. Within a class, a continuum of objects are available. The specification method is based on a mathematical property that is simple, intuitive and geometric.

We have completed a preliminary implementation of the method for generating fractals as sets of points in 2D. With this we have been able to easily produce many of the classic 2D deterministic fractals found in [Mand83]. Quickly finding an IFS for these sets was made possible because we have a method of solving the inverse problem, either approximately, or, in some cases, exactly. This means that given such an object, we can find the IFS that will compute it. One consequence of this is that new fractal sets, never before computed, have been generated which model (the projection of) trees, leaves and plants[1]. In fact, it is possible to approximate any set arbitrarily well with a (finite) IFS (although for some sets, better methods may be available).

As the complex objects of greatest interest to computer graphics are natural objects, we begin with a review of previous work in this area. We then present the mathematical basis of IFS followed by a description of our system and some examples of objects we have created.

2. Previous Techniques for Modeling Natural Objects

The differences between man-made and natural objects lead to the need for distinct modeling methodologies, as has been pointed out in [Four80] and elsewhere. Since man-made objects have smooth surfaces and a simple structure, manual specification of these objects in terms of polynomial-based sets is viable. In contrast, natural objects exhibit very irregular, non-smooth, highly complex geometries.

[1] Work of Diaconis and Shahshahani, performed independently, was highlighted in the August 3, 1984 issue of the journal *Science*. Their work is based on the same mathematics as ours; see [Diac84] for details.

The smoothness and static representations of traditional modeling methodologies is known to result in various storage and rendering difficulties [Four82]. But the lack of an automated scheme for the user to interactively specify the possibly hundreds or thousands of geometric primitives is probably the dominant limitation. These difficulties have led to the development of a number of alternative approaches, providing varying degrees of automation.

2.1. Texture Techniques

An effective technique for increasing realism but not geometry is color modulation. The idea is to modulate the color of a surface to simulate the way "high frequency" surface variations modulate reflected light. The first such application of this entailed "texture mapping" [Catm75]. Other examples include [Fu78] and [Gard84].

The deficiencies of this method, such as insensitivity to changes in illumination conditions, were identified and overcome by [Blin78]. Here, a "bump" function is mapped onto a polynomially defined surface. An important simplification is obtained by only perturbing the original surface normals. This method has proven to be very effective as long as the changes to the projection of the image due to the perturbations are imperceptible However, it cannot model complex geometries as would be required for trees and most plants.

2.2. Geometric Techniques

These limitations can be avoided by generating the complete geometry of the object rather than merely simulating it. Initial efforts at modeling natural phenomena using algorithmic generation rather than manual design relied upon intuitive methods rather than attempting to model the actual phenomena (e.g. mountains in [Csur79]). More recent work appears to be developing techniques that reflect more directly the processes which produce the phenomena. Examples of these include particle systems [Reev83], plant growing techniques [Smit84] and [Aono84], and fractional Brownian motion [Four80], [Four82] and [Mand83].

Particle systems use points as their basic geometric primitive. These points are moved through space over some time interval. The movement, as well as other properties, are determined by sampling stochastic processes. Either one or multiple samples are mapped to a single frame. This technique can be effective for modeling filamentous plants (as paths of particles). However, modeling non-filamentous objects/plants may prove to be more difficult. Also, the controllable parameters give only limited control over the shape of resulting objects (e.g., initial directions and distributions).

Non-stochastic techniques have also been used to grow plants and trees [Smit84] and [Aono84]. In both of these cases, a set of "growing rules" are applied successively in a way that corresponds very roughly to the actual growing processes in plants and trees. These techniques have

drawbacks in an interactive design environment, however. The grammars, which syntactically restrict the growing rules, are not typically specified geometrically; and there is not an obvious continuum of parameters, as there is, say, with linear transformations, or with control points used in parametric surfaces. The appropriate grammar for generating a given class of objects may not be obvious (this is likely to be the case). In addition, the techniques usually require a sophisticated "interpretation" step to actually generate the complete geometry.

2.3. Fractal Based Methods

The only examples of fractal objects being used to model natural phenomena are based on fractional Brownian motion [Mand83] and [Four82]. For example, this has proven very effective at modeling terrain, particularly mountains and coastlines.

The primary control that these techniques provide over the resulting object is by the value of a single parameter determining the dimension. Additional control is provided by scaling. In [Four82], this is in terms of the fractal perturbation relative to the size of polynomial surfaces. The presence of underlying polynomial primitives in this technique tends to suggest the traditional interactive design problems alluded to above. It is not clear, for instance, as to how one can easily extend this technique to the modeling of trees and plants.

3. Mathematical Foundations of Iterated Function Systems

In contrast to the above cited methods, we believe that modeling with IFS either avoids their limitations or at least has them to a lesser degree. We will now introduce those mathematical aspects of Iterated Function Systems which are relevant to their application in computer graphics.

3.1. Iterated Function Systems

The mathematical basis for the modeling techniques used in our research is the theory of Iterated Function Systems as set forth by Barnsley and Demko in [Barn84b]. For our applications, an IFS on d-dimensional Euclidean space will have two distinct components. The first is a finite set of affine mappings of d-space into itself, $M=\{M_1,M_2,...,M_n\}$. The second component is a set $P=\{P_1,...,P_n\}$ of probabilities, where P can be thought of as relative weights for each of the maps and $\sum_{i=1}^{n} P_i=1$.

The mappings are used to define a random walk in d-space in the following way. When we are at the point z_0, we randomly choose a map, M_i, and move to $z_1=M_i(z_0)$. We then make another random choice of M_j, and move to $z_2=M_j(z_1)$. This continues indefinitely. In this process, the probability of choosing the map M_i is P_i. A necessary restriction on the maps is that they are strict contractions in the sense that their eigenvalues all have modulus less than one. Without this assumption the random walk might wander off to infinity.

Associated with every IFS is a unique compact set (i.e. closed and bounded) called the *attractor* of the IFS, and a special probability measure called the *P-Balanced measure* of the IFS which "lives" on the attractor, (the P refers to the set of probabilities). Intuitively, the attractor is the set about which the random walk eventually clusters. The P-Balanced measure quantifies this clustering by ascribing a sense of density, or height, to the attractor.

If you start your walk away from the attractor, you will be drawn to it with the rate of attraction being determined by the eigenvalues of the maps. Once you land on the attractor (you may begin there), you will never leave it; and the amount of time you spend in any particular part of the attractor is given exactly by the P-Balanced measure. The geometry of the attractor in d-space is dependent only on M, and so is independent of P. However, P affects the P-Balanced measure. The P-Balanced measure is the *stationary distribution* of the random walk. Its *support* is the attractor.

3.2. An Example: the Cantor Set

The two functions $f(x)=\frac{x}{3}$ and $g(x)=\frac{x}{3}+\frac{2}{3}$ together with $P=\{0.5,0.5\}$ form an IFS on the real line (if the numbers are complex, then this is an IFS in the complex plane). If we choose 2 as our starting point on the line, and if the first four steps of our random walk are determined by the selection of the mappings f, f, g, f in that order, then we will visit the points $\frac{2}{3}$, $\frac{2}{9}$, $\frac{20}{27}$, and $\frac{20}{81}$. As it turns out, no matter where we start, our random walk will take us closer and closer to the well known Cantor set pictured below. This then is the attractor for the IFS.

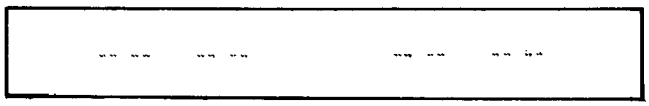

Figure 3.1

The significance of P being $\{0.5,0.5\}$ is that on our walk we will spend as much time on the left half of the Cantor set as on the right. Moreover, half of the time we spend on the left half will be spent on the left half of it, and so on. In short, all portions of the Cantor set are visited with a frequency proportional to their relative size. If we had skewed the probabilities by taking $P=\{0.1,0.9\}$, then only one tenth of our time would be spent on the left side of the Cantor set, and of that time only one tenth would be spent on the left half of the left half, etc.

Hence, the inclusion of the probabilities allows us to associate a probability distribution or measure with our random walk, and thereby contributes added structure to it. This is the P-Balanced measure, which is a countably

additive, Borel measure of total mass one. We can think of the measure in terms of a histogram giving relative percentages of time spent in various parts of the Cantor set.

3.3. Construction of Some Well Known Fractals by IFS

Iterated Function Systems provide a general context for the construction of a wide variety of geometric objects, including fractals. Viewing fractals as attractors of IFS not only gives a completely new way to construct them, but also makes it possible to study a particular fractal in a larger context. For example, Barnsley and Harrington [BaHa84] studied the attractors of the 2-space IFS given by $M_1(z)=sz+1$ and $M_2(z)=sz-1$ with $P=\{0.5,0.5\}$. Here z is a complex variable and s a complex parameter. For each value of s, we have a different IFS.

Their work focused on certain topological aspects of the attractor as a function of s. With $s=\frac{1}{3}$ the attractor is the Cantor set. With $s=\frac{i}{2}+\frac{1}{2}$ the attractor is the dragon curve encountered by many people, cf. [Mand83]. Our construction of this shape (Figure 3.2) was accomplished with this IFS, as opposed to the space filling curve method of [Mand83]. Generally, one can watch the shape of the attractor change continuously as the parameter s is varied. In this way, we can see that, for instance, the dragon curve fits into a family of fractals, each determined by a single complex number.

The classic Sierpinski gasket and Koch island fractals also have simple IFS descriptions. For example, to construct the gasket, choose three non-colinear points in the plane. For each point construct the affine map that keeps the given point fixed and attracts all of 2-space to that point, with the attraction factor (scaling) being 0.5 in all directions. Figure 3.3 shows the resulting attractor.

3.4. Constructive Aspects of the Attractor and the Inverse Problem

A very important property of the attractor is that it is the set theoretic union of the images of itself, under the functions in the IFS. That is, if A denotes the attractor, then

$$A = M_1(A) \bigcup M_2(A) \bigcup \cdots \bigcup M_n(A). \qquad (3.1)$$

This "self covering" property holds the key to the approximation or "recovery" of fractal sets by attractors of IFS. Utilization of this property in a modeling system will be discussed in Section 4. In addition, approximations to the classical moments of the P-Balanced measure can be obtained from a digitized image. These can then be used in the construction of attractors that will approximate the image. This theory is worked out in [Barn84b] and is also discussed in [Diac84].

3.5. Qualitative Aspects of Attractors

We also have the capacity to construct attractors having prescribed bounds on their Hausdorff dimension, or having prescribed subsets. The attractor of an IFS is generally a fractal set (although it does not have to be one). In the case that the images of the attractor under the maps in the IFS are mutually disjoint or just touch, the Hausdorff dimension can be bounded from above and below in terms of the eigenvalues of the maps [Barn84]. This provides for some control over the dimension of the attractor. For example, if there are n maps, each with the same symmetric scaling factor S, then the dimension is exactly $\frac{\ln n}{\ln \frac{1}{S}}$.

One use of this is in the construction of fractal interpolants of data. For example, if we are given a finite number of points in the plane, we know how to construct a fractal curve that passes through these points. Each curve segment between two points will have global properties inherited from the entire set of interpolants. Thus, this differs from the interpolation by fractional Brownian motion in [Four82], where each interpolation between consecutive sample points is independent of the other sample points.

This is accomplished by forcing the linear parts of the maps (upper left 2x2 sub-matrix of homogeneous matrices) to all have a common eigenvector. Typically, the maps are chosen so that the y-axis is an eigenvector. By adjusting the eigenvalue (scaling factor) of each map, we can force the Hausdorff dimension to lie in a prescribed range, and thus control the "thickness" of the curve. Figure 3.4 illustrates this technique. Both curves interpolate the same data but have different Hausdorff dimensions. The technical details can be found in [Demk85] as well as in [Barn85], where extensions to higher dimensional interpolation are discussed.

A very useful device which allows us to have any preassigned shape in the attractor is the addition of a set-valued mapping C_B to the IFS. The set B is called a "condensation set". The new mapping C_B has the property that for any point z in d-space, $C_B(z)=B$. For example, if we add the map $C_{\{0.5\}}$ to the IFS that gave us the Cantor set, then the new attractor will contain the point 0.5, as well as all images of 0.5 under all iterates of the maps in the IFS. The attractor in this condensation example is not the Cantor set, but in some sense is built on it, and near 0 and 1 will look a great deal like the Cantor set. A simple use of condensation for geometric modeling is demonstrated at the end of Section 4.

4. Geometric Modeling with IFS

We now examine our current application of the theory of Iterated Function Systems to geometric modeling. Our work to date has been limited to two-dimensional sets.

4.1. Designing

In Figure 4.1, we have an image of a maple leaf that is the attractor of an IFS with four maps. We will now discuss how one could have ascertained those maps, i.e. how we could solve the inverse problem. Let us assume that we have some initial idea of the object we wish to model. The intuitive key for deriving an IFS that will model any given object is the notion of *self-tiling*.

One can always view an object as the union of several sub-objects. Consider that the sub-objects are actually instances of the original object. In particular, each sub-object is obtained by applying an affine transformation to the entire object. We will call each of these sub-objects a *tile*. Now we can think of *tiling* the original object with two or more affinely transformed copies of itself. The tiling scheme should completely cover the object, even if this necessitates overlapping the tiles.

We now have enough information to determine an IFS whose attractor will be this object. In particular, each transformation used to "create" a tile corresponds exactly to one map in the IFS. If the self-tiling is exact, the IFS can be used to generate exactly that object. Otherwise, the attractor will approximate the object. Note that the notion of *sub-object* implies that the absolute values of the scaling factors are strictly less than one, which was stated in Section 3 as a restriction on the modulus of the eigenvalues of the maps.

Figure 4.2 shows the same leaf as in Figure 4.1, but each tile is drawn in a separate color. Each of the sub-views of Figure 4.2 illustrates the effect of deleting one map from the IFS of the leaf. The left and right upper views show deleting the lower and upper central maps, respectively. The views below the primary leaf demonstrate omitting the right and left maps. Figure 4.3 shows the tiles of the dragon curve in Figure 3.2 and Figure 4.4 is Sierpinski's gasket as in Figure 3.3.

All of the foregoing discussion is in fact simply a restatement of Eq. 3.1. However, this is a somewhat remarkable statement. What it means is that these objects are defined in terms of themselves: they are self-referential. The classic problem of circular definitions is circumvented by describing the object by a "meta-system", i.e. the set of maps forming the IFS.

4.2. Computation of the Attractor

Let us assume that our geometric primitive upon which we will iterate is a point, as is the case in all of the pictures in this paper. We choose a map M_i at random with probability P_i and transform the point by that map. We then draw the resulting point. Next, another random selection is made, and that map is applied to the result of the previous transformation (as opposed to the original point). We then draw this second point. This process continues for some user-defined number of iterations, always applying a randomly selected map to the most recently resulting iterate.

Two criteria can be used for selecting the initial point. If one wants only points on the attractor, the initial point must be on the attractor itself. This is sufficient since once the iteration lands on the attractor, it will produce only points that lie on the attractor. Since all fixed points are on the attractor, one of these is an obvious choice. An alternative that is useful for special effects is to select an initial point reasonably far from the attractor. A path from this point to the attractor will then be generated. Using a set of initial points produces the effect illustrated in Figure 4.5.

The set of points that results is not dependent simply upon the maps. Although the attractor is fixed by the IFS, only a finite subset can be computed. Two ways are available for controlling which subset is generated. The most obvious is simply selecting the total number of points/iterations produced. The second source of control is the weightings/probabilities. The relative density of the tiles will be exactly their relative weights, i.e their probability of being selected each time. In fact, the probability of plotting any given point/pixel is governed by the P-Balanced measure. Thus, it is possible to control directly the distribution of points in the representation. This is very convenient for design and efficiency of computation.

We have also implemented a deterministic method of computation which produces all possible combinations of iterations of finite length. Space limitations preclude its presentation here, but the interested reader may find its description, as well as a number of other details, in [Demk85].

4.3. Other System Aspects

Specification of the components of a map are given in terms of differential scaling, differential rotation, and either a translation vector or a fixed point about which scaling and rotation occur. Differential rotation is specified in terms of an angle to rotate the x-axis and a separate angle to rotate the y-axis. The probabilities are specified as relative (integer) weights, thus freeing the user from being concerned with the normalizing required to guarantee that they sum to one.

One useful coloring scheme is to associate a color with each map and color the iterate according to the last transformation applied to produce it. This results in revealing the tiling as demonstrated in Figure 4.2 and can be an important aid in interactive design. Another scheme is to change the color of a pixel every time a point is mapped onto it. The color variations then correspond to the P-Balanced measure of the set. This was done in Figure 4.3. This can be used, for example, for special shading effects.

So far we have only discussed using a point primitive in the generation of an image. In fact, any primitive can be used. This is likely to be important in making effective uses of IFS. Currently we have a limited implementation that iterates on a set of line segments. This provides the capability of generating sets similar to those of [Smit84]

and [Aono84] that use grammars. We anticipate extending this to a broader class of geometric primitives. One interesting possibility would be to iterate on a set that is itself a fractal.

We close this section with pictures of a fern (Figures 4.6 and 4.7). This was actually constructed by examining a botanical drawing of a Black Spleenwort Fern and constructing a self-tiling for it. Figure 4.7 shows a close up, and demonstrates the ability to produce as much detail as needed. It also illustrates the use of condensation introduced in Section 3.5. The stems were produced by introducing a condensation set represented by a single line segment.

5. Future Work

The exploration and application of Iterated Functions Systems are in their infancy, but initial results are quite exciting. Currently there is no other comparable methodology for producing the range of complex images with so little user specification (and storage). Experience with our system has shown that one has considerable control that permits fine tuning of the structure, and the range of significantly different objects easily producible is impressive. The fact that the approach rests upon a firm mathematical basis gives us confidence in being able to find many ways of solving various geometric modeling problems.

The pictures included in this paper represent the simplest application of this technique. For real objects, they depict only a kind of silhouette, but this in no way represents the limits of the technique. The extension of our methods to three dimensional objects is the most important next step. Since the set of transformations determining an IFS can be defined for any dimension, generating an attractor in 3D is straightforward.

As discussed in Section 4, the primitive that is iterated need not be restricted to points; any set can be iterated. Therefore, a more viable approach may be to iterate on surfaces. Also the incorporation of stochastic techniques for specifying the maps is expected to lead to even more realistic images than are currently possible. In addition to representational and computational issues, the interactive design problems must be addressed. The self-tiling property provides a very important aid, but we have discerned other properties, only a few of which are understood in a formal way.

Acknowledgements

We would like to acknowledge the considerable contribution of Michael Barnsley to the development of the theory of IFS, and his valued assistance in helping with its application to geometric modeling. The IFS for the Black Spleenwort Fern is also due to his efforts. Special thanks go to Bill Thibault for his help with the photographic equipment.

References

[Aono84]
Aono, Masaki and Tosiyasu L. Kunii, "Botanical Tree Image Generation", *IEEE Computer Graphics and Applications,* 4 (5), pp.10-33, (May 1984).

[BaHa84]
Barnsley, Michael F. and Andrew N. Harrington, "A Mandelbrot Set for Pairs of Linear Maps" to appear in *Physica* 14D, (1985).

[Barn84a]
Barnsley, Michael F. and Stephen Demko, "Rational Approximation and Interpolation", *Proceedings of the Tampa Conference on Rational Approximation,* edited by P.R. Graves-Morris, E.B. Saff, and R.S. Varga, Springer Verlag Lecture Notes in Mathematics, No. 1105, pp. 73-88 (1984).

[Barn84b]
Barnsley, Michael F. and Stephen Demko, "Iterated Function Systems and the Global Construction of Fractals", to appear in *The Proceedings of the Royal Society,* preprint available, School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332.

[Barn85]
Barnsley, Michael F., "Fractal Interpolation", preprint available, School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332.

[BEHL84]
Barnsley, Michael F., Vincent Ervin, Doug Hardin, and John Lancaster, "Solution of an Inverse Problem for Fractals and Other Sets", preprint available, School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia, 30332.

[Blin78]
Blinn, James F., "Simulation of Wrinkled Surfaces", *Computer Graphics* **12** (3), pp. 286-292 (Aug. 1978). SIGGRAPH '78 Proceedings.

[Catm75]
Catmull, Ed, "Computer Display of Curved Surfaces", *Proc. IEEE Conference on Computer Graphics, Pattern Recognition and Data Structure,* (May 1975).

[Csur79]
Csuri, C., R. Hackathorn, R. Parent, W. Carlson, and M. Howard, "Towards an Interactive High Visual Complexity Animation System," *Computer Graphics* **13** (2), pp. 289-299 (Aug. 1979). SIGGRAPH '79 Proceedings.

[Demk85]
Demko, Stephen, Laurie Hodges, and Bruce Naylor, "Application of Iterated Function Systems to Geometric Modeling", Technical Report GIT-ICS 85/14.

[Diac84]
Diaconis, Persi and M. Shahshahani, "Products of Random Matrices and Computer Image Generation", Stanford University preprint.

[Four80]
Fournier, Alain, "Stochastic Modeling in Computer Graphics", PhD thesis, Univ. of Texas at Dallas, Richardson, Texas, (Aug. 1980).

[Four82]
Fournier, Alain, Don Fussell, and Loren Carpenter, "Computer Rendering of Stochastic Models", *Communications of the ACM,* **25** (6) (June 1982).

[Fu78_]
Fu, K.S. and S.Y. Lu, "Computer Generation of Texture Using a Syntactic Approach", *Computer Graphics* **12** (3), pp. 147-152 (Aug. 1978). SIGGRAPH '78 Proceedings.

[Gard84]
Gardner, Geoffrey Y., "Simulation of Natural Scenes Using Quadric Surfaces", *Computer Graphics* **18** (3), pp.11-20 (July 1984). SIGGRAPH '84 Proceedings.

[Hutc81]
John Hutchinson, "Fractals and Self-similarity", *Indiana University Journal of Mathematics,* **30** , pp. 713-747 (1981).

[Mand83]
Mandelbrot, Benoit, *The Fractal Geometry of Nature* W. H. Freeman and Co., San Francisco, (1983).

[Reev83]
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).

[Smit84]
Smith, Alvy Ray, "Plants, Fractals, and Formal Languages", *Computer Graphics* **18** (3), pp. 1-10 (July 1984). SIGGRAPH '84 Proceedings.

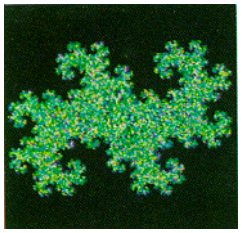

Figure 3.2
Dragon Curve with P-Balanced Measure

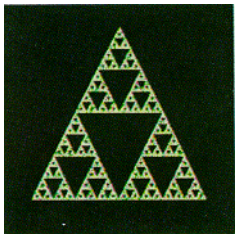

Figure 3.3
Sierpinski's Gasket

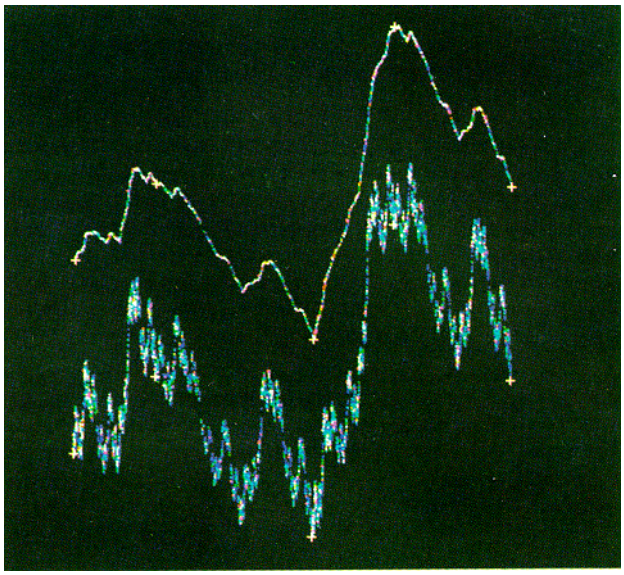

Figure 3.4
Fractal Interpolation

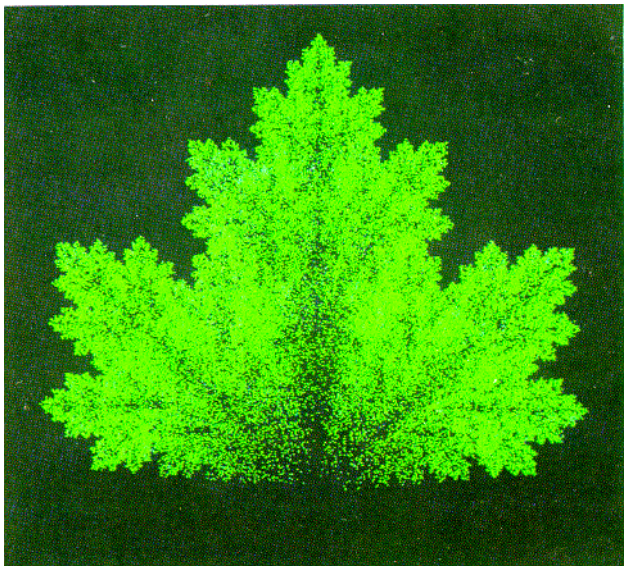

Figure 4.1
Maple Leaf

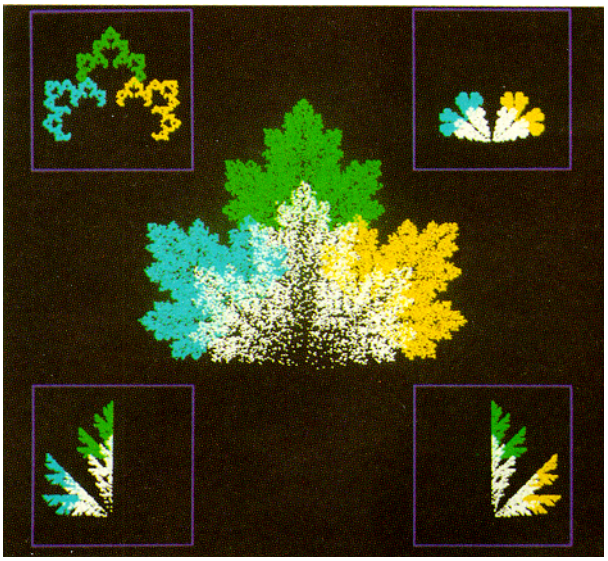

Figure 4.2
Tiled Maple Leaf with Subsets of the IFS

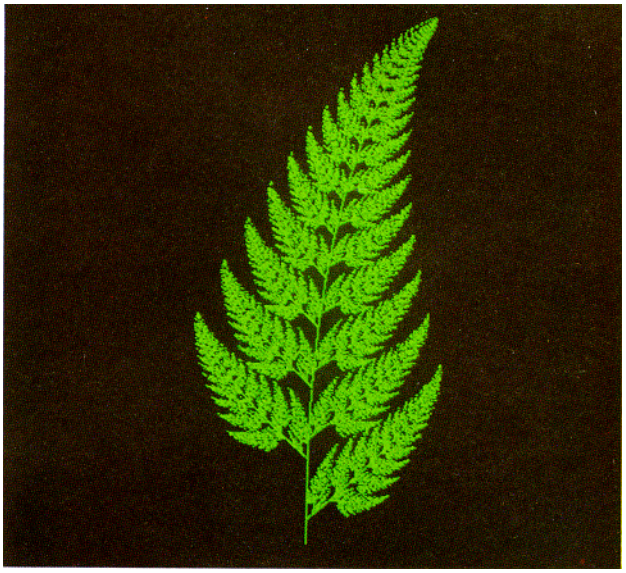

Figure 4.6
Black Spleenwort Fern

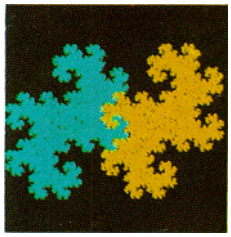

Figure 4.3
Tiled Dragon Curve

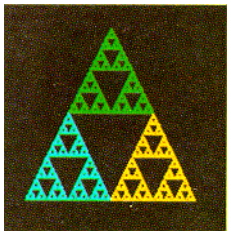

Figure 4.4
Tiled Sierpinski's Gasket

Figure 4.7
Close-up of Black Spleenwort Fern

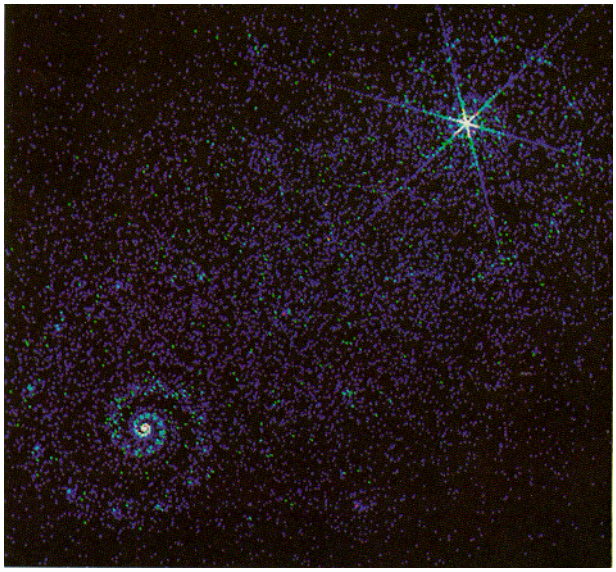

Figure 4.5
Galaxy image produced by plotting random walks