

Rendering Methods for Iterated Function Systems

Daryl Hepting and Przemyslaw Prusinkiewicz

Department of Computer Science

University of Regina

Regina, Saskatchewan, Canada

Dietmar Saupe

Institut für Dynamische Systeme

Fachbereich Mathematik und Informatik

Universität Bremen

2800 Bremen 33, West Germany

This paper describes rendering methods for iterated function systems (IFS's). The rendering process consists of the generation of a field of data using an IFS and its visualization by means of computer graphics. Two groups of methods are presented: 1. Rendering of the attractor \mathcal{A} of an IFS. These *attracting* methods may visualize the geometry and additionally the invariant measure supported by the attractor. 2. Rendering the complement of the attractor. There are three approaches, namely methods representing *Euclidean distance* from \mathcal{A} ; *repelling* methods, computing the escape time of a point from \mathcal{A} ; and methods using (electrostatic) *potential functions* of the attractor. The last of these methods calculates integrals with respect to the invariant measure of the attractor. An algorithm which generates an approximation of such integrals with prescribed tolerance is presented. This provides an alternative to the usual approach based on Elton's ergodic theorem and time averages of trajectories generated by the "chaos game", where no error bound is available. Algorithms specifying the details of all methods are presented, some of them in the form of pseudocode. Examples of images obtained using these algorithms are given. The relationship to previously developed methods for visualizing Mandelbrot and Julia sets is also discussed.

1 Introduction

One fascinating aspect of fractals is the beauty of their graphical representations. In this paper we consider methods representing fractals that are specified using iterated function systems (IFS's) [1,2,10]. They are diagrammatically represented in Figure 1 and compared to analogous methods for rendering Julia sets. In fact, the main motivation of

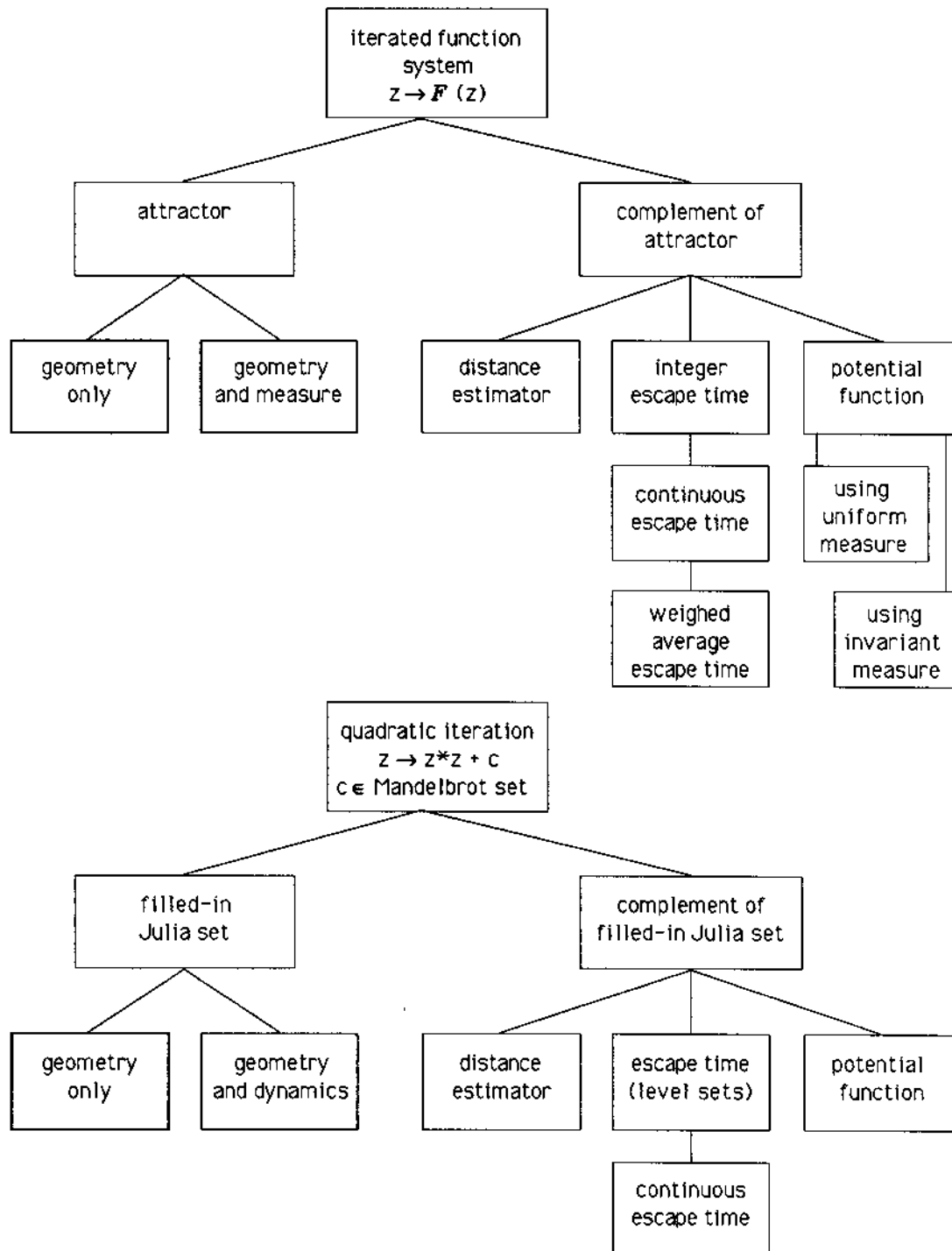


Figure 1: An overview of the methods for rendering iterated function systems and their correspondence to methods for Julia sets. For clarity the methods for rendering Julia sets are subdivided into those for the filled-in Julia sets and those for the corresponding complements. In a particular picture both aspects are usually combined. In place of a series of images we give a reference for each of the cases. See [13, Fig. 3.7] (geometry only), [12, Fig. 27, Map 25] (geometry and dynamics), [13, Plate 28, 29] (distance estimator, here for the Mandelbrot set), [12, Map 18] (level sets) and [13, Plate 27] (continuous escape time and potential function). The last two methods are equivalent, the escape time is proportional to the logarithm of the potential function. For iterated function systems analogous methods are split into those for the attractors and those for their complements. It is shown in Section 4 that escape time and distance are equivalent within a certain class of IFS's.

this paper was to establish this strong correspondence between methods for Julia sets and IFS's, thus extending the approach initiated in [14,15]. The paper is organized as follows. In the next section attention is focused on the *attractive* rendering methods. They make direct use of the definition stating that the set specified by an IFS $\mathcal{F} = \{F_1, F_2, \dots, F_N\}$ is the smallest set \mathcal{A} , closed in the topological sense, such that the image of any point of \mathcal{A} under any of the functions $F_i \in \mathcal{F}$ also belongs to \mathcal{A} . Starting from a point $z_0 \in \mathcal{A}$, one can approximate \mathcal{A} as a set of images of z_0 under compositions of transformations from \mathcal{F} . It is usually assumed that all functions of \mathcal{F} are contractive. Consequently, if the starting point z_0 does not belong to \mathcal{A} , the consecutive images of z_0 gradually approach \mathcal{A} . For that reason, the set \mathcal{A} is called the *attractor* of the IFS \mathcal{F} , and methods of rendering it based on finding the images $F_{i_k}(F_{i_{k-1}}(\dots(F_{i_1}(z_0))\dots))$ are termed attractive methods. We present a classification of these methods and introduce a deterministic algorithm that provides a uniform distribution of the points approximating the attractor, provided that the sets $F_i(\mathcal{A})$ are pairwise disjoint or "just touching" (they don't overlap). This is a deterministic analogue of the probabilistic approach, termed the *chaos game*, in which *probabilities* attached to the affine maps are carefully chosen in order to achieve a uniform distribution of points over the attractor [2]. These probabilities or *weights* may sometimes be chosen with a different motivation in mind, namely such that the chaos game produces a point distribution corresponding to the *invariant measure* (see below). The density function of such a distribution can be interpreted as color, resulting in a shaded image of the attractor. We show that the deterministic method can be readily modified for this purpose as well.

The attracting methods make it possible to approximate the attractor defined by an iterated function system, but do not provide any information about points outside of the attractor. The experience with the Mandelbrot set shows that beautiful, intriguing images can be obtained using properties of the points outside of the attractor [12,13]. We extend this approach to iterated function systems.

In Section 3 we consider three methods based on the *Euclidean distance* from the set. The first method estimates distance from the attractor \mathcal{A} as the minimum distance between a given sampling point and a finite set of points approximating \mathcal{A} . The second method improves the computation time and reduces space requirements by dividing recursively the attractor into regions. Those regions that lie farther than the closest attractor point already found are removed from further consideration. The third method again reduces the distance evaluation time by exploiting the continuity of the distance function.

In Section 4 we consider another group of methods for visualizing regions outside the attractor. We call them *escape time* or *repelling methods*, since they evaluate the escape time of a sampling point, repelled from the attractor by application of the inverse transformations of the IFS. In contrast to previously published methods which resulted only in integer values for the escape time [2,15], the methods considered in this paper compute real-valued continuous functions. Three methods are presented in detail. The first one calculates the escape time for a given sampling point directly from the definition. The second method evaluates the escape-time function in an array of sampling points rather than in a single sampling point. Previously computed escape time values are used to accelerate the computation for other grid points. The third method uses point-to-point coherence to update the tree of transformations.

It is intuitively clear that the closer the points are to the attractor, the larger escape times they have. We show that this intuition is true in a much stronger sense, i.e. in some cases the distance-based and escape-time algorithms are in a certain way equivalent (Section 4.6).

Our final method presented in Section 5 visualizes a potential function of the attractor. It can be conceptualized by the *electrostatic potential* that arises when the attractor is charged with electricity and placed in an otherwise void (chargeless) space. We discuss two ways to distribute the charge on the attractor: uniform distribution, and secondly, distribution according to the invariant measure of the attractor. Again, several implementations with differing degrees of efficiency are described.

We conclude the introduction by listing the definitions for iterated function systems as used in this paper.

Let X be a complete metric space with distance function d . An iterated function system is defined as a finite set $\mathcal{F} = \{F_1, \dots, F_N\}$ of transformations $F_i : X \rightarrow X$ such that for all $i = 1, \dots, N$ the following inequality holds:

$$\rho(F_i) = \inf_{x \neq y} \frac{d(F_i(x), F_i(y))}{d(x, y)} < 1 .$$

The constants $\rho(F_i)$ are called *contraction factors* or contraction ratios. In this paper the metric space is always the Euclidean space \mathbf{R}^n , typically \mathbf{R}^2 , equipped with the Euclidean norm $\|\cdot\|$, and the mappings F_i are *affine contractions*, that is

$$F_i(x) = A_i x + B_i$$

where A_i is a square matrix with n rows and B_i is a vector with n elements.

Each iterated function system implicitly defines a compact set \mathcal{A} , called the *attractor* of the IFS, which is the unique (nonempty) solution of the equation [10]

$$\mathcal{A} = \bigcup_{i=1}^N F_i(\mathcal{A}) . \quad (1)$$

An important notion related to the sets defined by IFS's is that of a measure, which, informally speaking, can be interpreted as the distribution of points over the attractor. Specifically, there are so called *invariant measures* μ which satisfy an equation corresponding to $\mathcal{A} = \bigcup F_i(\mathcal{A})$, namely

$$\mu = \sum_{i=1}^N p_i \mu F_i^{-1}$$

where p_1, \dots, p_N are weights or probabilities associated with the transformations F_1, \dots, F_N such that $p_1 + \dots + p_N = 1$ [10]. According to the ergodic theorem of Elton [6], the invariant measure can be approximated almost surely by the distribution of points generated by the chaos game, described in the next section.

Given the invariant measure μ one can consider integrals of continuous functions defined over the attractor, $\int f d\mu$. The evaluation of this integral can proceed using the chaos game and the ergodic theorem. In Section 5.2 we provide an alternative deterministic method.

The strict condition on the contraction factors can be relaxed to an *average contractivity condition*, and some of the methods presented in this paper can be modified to suit that situation. However, for simplicity our discussion is limited to the case of strict contractions stated above. See [2] for a comprehensive introduction to IFS theory.

2 Attracting rendering methods

Let us assume that we know a starting point z_0 that belongs to the attractor \mathcal{A} . From equation (1) it follows that its images, namely $F_1(z_0), \dots, F_N(z_0)$, also belong to \mathcal{A} . By applying the same argument again we conclude that all images of the form¹ $F_i F_j(z_0)$, where $i, j = 1, \dots, N$, belong to \mathcal{A} as well. In general,

$$F_{i_1} \cdots F_{i_n}(z_0) \in \mathcal{A}$$

with $i_1, \dots, i_n = 1, \dots, N$. The attracting methods for IFS's follow directly from this observation and operate by:

- selecting a starting point,
- applying to this point the tree of sequences of functions $F_i \in \mathcal{F}$ (Figure 2).

Depending on the order of applying transformations F_i and the way the results of previous iteration steps are used, the attractive methods can be divided into *breadth-first*, *depth-first* and *stochastic*. The difference between breadth-first and depth-first methods lies in the order of constructing the tree of transformations. In the breadth-first case, all transformation sequences of length i will be constructed before the first sequence of length $i + 1$ is considered. In contrast, according to the depth-first approach, a sequence of transformations will be extended to its maximum depth before other sequences are formed. The stochastic approach (chaos game) is different from the preceding two in that only one path in the tree of images is explored. In each step, a transformation is chosen in a stochastic way. The sequence of points constructed this way spreads over the entire set \mathcal{A} without leaving any portions uncovered. From this viewpoint, the chaos game is simply one of several methods for generating the attractor \mathcal{A} and not an inherent part of the IFS definition.

Orthogonal with respect to the above classification is a partitioning of attractive methods into those using *postmultiplication* and those using *premultiplication* of transformations.

In both cases a sequence of points z_1, z_2, \dots is computed based on an initial point z_0 and a sequence of transformations $F_{i_1}, F_{i_2}, \dots \in \mathcal{F}$. In the postmultiplication case the points z_k are given by

$$z_k = F_{i_k} \cdots F_{i_1}(z_0)$$

i.e.

$$z_{k+1} = F_{i_{k+1}}(z_k), \quad k = 0, 1, \dots$$

¹Throughout this paper we use the convention that composite mappings are applied right to left, i.e. for example $fgh(x) = f(g(h(x)))$.

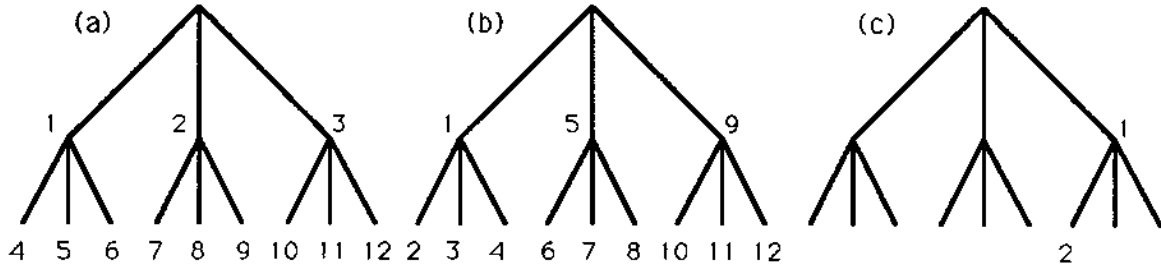


Figure 2: The order of the computation of points in the tree of images for (a) breadth-first, (b) depth-first and (c) stochastic methods.

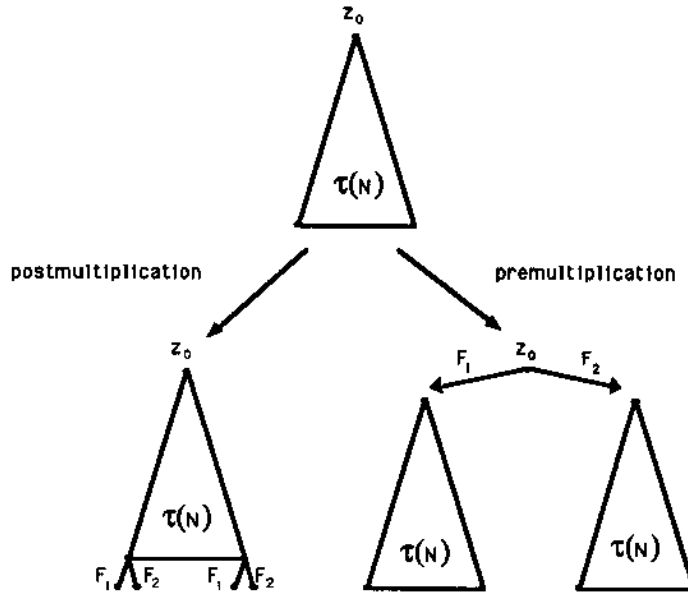


Figure 3: Extension $\tau(n + 1)$ of the transformation tree $\tau(n)$ in the postmultiplication and premultiplication cases. In the first case (left) the tree $T(n)$ is extended by applying transformations $F_{i_{n+1}}$ to the points $z' = F_{i_n}(F_{i_{n-1}}(\dots(F_{i_1}(z_0))\dots))$, which are the leaves of tree $T(n)$. The coordinates of the points z' must be known, but the composite transformations $F_{i_1}F_{i_2}\dots F_{i_n}$ are not needed. In the premultiplication case (right) the extension of transformation sequences can be viewed as an application of the existing tree of transformation $T(n)$ to the set of starting points $F_1(z_0), \dots, F_N(z_0)$. Consequently, the transformations associated with different paths within $T(n)$ (terminating on tree leaves) must be known.

while in the method that uses premultiplication we have

$$z_k = F_{i_1} \dots F_{i_k}(z_0)$$

and the relation between z_k and z_{k+1} is more complicated, namely

$$z_{k+1} = F_{i_1} \dots F_{i_k} F_{i_{k+1}} F_{i_k}^{-1} \dots F_{i_1}^{-1}(z_k)$$

assuming the invertibility of all transformations $F_i \in \mathcal{F}$. In the first case the generated sequence of points may wander about the attractor in an apparently random fashion, while in the other case the point sequence always converges to a particular point in \mathcal{A} which is determined by the choice of transformations F_{i_1}, F_{i_2}, \dots

One consequence of this observation is that the stochastic method cannot be used in conjunction with premultiplication of transformations. The other two methods (breadth-first and depth-first) for finding consecutive points of the attractor can be implemented

using either premultiplication or postmultiplication of transformations. The properties of the resulting methods can be characterized as follows:

- The breadth-first and stochastic methods can be run without specifying in advance the desired accuracy of the approximation (for example, in terms of the tree depth or the total number of points plotted). The generation of points can be stopped interactively at any time, after the desired accuracy in the rendering of the attractor has been reached. In contrast, the depth-first methods require a predefined criterion for terminating the recursion.
- Noting by N the total number of points in the approximation, and assuming that the tree of transformations is balanced, the breadth-first algorithms require $O(N)$ space, the depth-first algorithms require $O(\log(N))$ space, and the stochastic algorithms require constant space.
- The depth-first method with premultiplication constructs the attractor part by part, with final accuracy. The remaining methods construct approximation of the entire attractor with a gradually increasing accuracy.

A comparison of the order of drawing points in the attractor rendered using the depth-first methods is shown in Figure 4. The complete image (a) contains $N = 29,524$ points, and corresponds to a balanced tree of depth 9. This image is the same for the premultiplication and postmultiplication methods. Image (b) shows the postmultiplication method in progress, after $\frac{2}{9}N$ points have been drawn. Images (c) and (d) show the premultiplication method in progress, with $\frac{2}{9}N$ and $\frac{1}{2}N$ points drawn.

If all mappings that define an attractor have the same contraction ratio and the regions $F_i(\mathcal{A})$ do not overlap, an approximation of the attractor using a balanced transformation tree exhibits uniform point distribution. On the other hand, if the contraction ratios differ from each other, then the use of a balanced tree results in a nonuniform point distribution. The chaos game attempts to provide a uniform covering of the attractor in spite of unequal contraction ratios (see e.g. [2]). Below we present a deterministic method which serves the same purpose. This is a version based on ideas first given by Williams [18], see also [5]. The idea is to cut branches of the transformation tree after the desired accuracy of the approximation ϵ has been reached. It corresponds to the modified inverse iteration method for rendering Julia sets described in [13, page 178]. We illustrate the approach using depth-first technique with premultiplication of transformations.

2.1 The adaptive-cut algorithm

For $\epsilon > 0$ we construct a point set which has a Hausdorff distance less than ϵ from the attractor \mathcal{A} of the IFS. We recall that the Hausdorff distance between two non-empty compact sets A and B is defined as

$$d_H(A, B) = \max \left\{ \sup_{x \in A} \inf_{y \in B} \|x - y\|, \sup_{y \in B} \inf_{x \in A} \|x - y\| \right\}.$$

Let $\rho_k < 1, k = 1, \dots, N$ be the contraction ratios for the affine maps F_1, \dots, F_N of the IFS. The sets $F_k(\mathcal{A})$ cover \mathcal{A} , $\mathcal{A} = \bigcup_{k=1}^N F_k(\mathcal{A})$. If we let

$$R = \text{diam}(\mathcal{A})$$

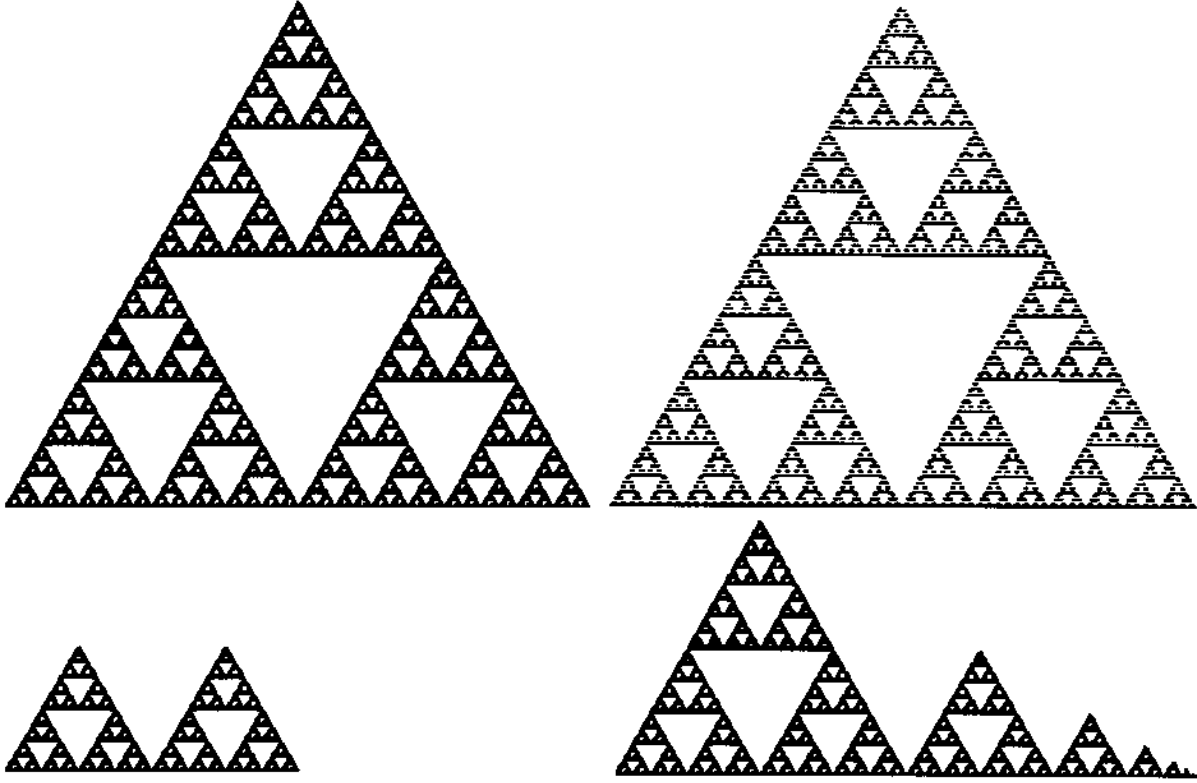


Figure 4: A comparison of the order of drawing points in the attractor. (a) The final image. (b) Partial image obtained using depth-first algorithm with postmultiplication. (c,d) Partial images obtained using depth-first algorithm with premultiplication.

denote the diameter of \mathcal{A} , then the covering sets $F_k(\mathcal{A})$ have diameters not exceeding $\rho_k R$. Each set $F_k(\mathcal{A})$ is in turn covered by sets $F_k(F_l(\mathcal{A}))$, $l = 1, \dots, N$, having diameters less than or equal to $\rho(F_k F_l)R < \rho_k \rho_l R$, and so forth. Thus, since the contraction ratios ρ_k are less than 1, it is possible to cover the whole attractor \mathcal{A} by sets of diameter not exceeding ε . These covering sets are given in the form

$$F_{k_1} F_{k_2} \cdots F_{k_n}(\mathcal{A})$$

with

$$\rho(F_{k_1} F_{k_2} \cdots F_{k_n}) \leq \frac{\varepsilon}{R}.$$

The above concept is illustrated in Figure 5. The IFS consists of four transformations $\{F_1, \dots, F_4\}$. Each transformation maps the attractor \mathcal{A} (large square) onto a smaller square or rectangle which shares a corner with \mathcal{A} (Figure 5a). The square $F_1(\mathcal{A})$, located near the bottom left corner of \mathcal{A} , has the diameter equal to the accuracy ε , thus it is not subdivided further in the process of approximating \mathcal{A} . The remaining regions require a further subdivision for one or two more steps before their diameter falls below ε (Figure 5b).

In order to approximate the attractor (\mathcal{A}) with accuracy ε , it is sufficient to pick a starting point $z_0 \in \mathcal{A}$ and apply the composite transformations corresponding to each leaf of the transformation tree. The result is a set $\mathcal{A}_\varepsilon \subset \mathcal{A}$, thus

$$\sup_{x \in \mathcal{A}_\varepsilon} \inf_{y \in \mathcal{A}} \|x - y\| = 0.$$

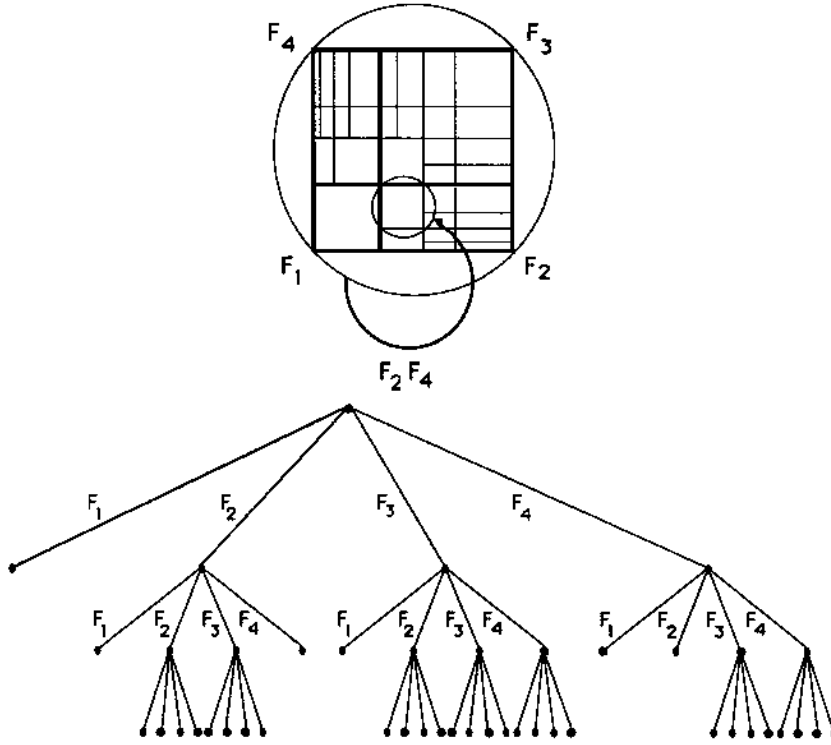


Figure 5: (a) Example of the covering sets $F_{k_1} F_{k_2} \cdots F_{k_n}(\mathcal{A})$, and (b) the resulting tree of transformations with the branches cut using the contraction ratio criterion.

By construction, for each point $y \in \mathcal{A}$ there exists a point $x \in \mathcal{A}_\varepsilon$ within the distance $\|x - y\| \leq \varepsilon$. Consequently,

$$\sup_{y \in \mathcal{A}} \inf_{x \in \mathcal{A}_\varepsilon} \|x - y\| \leq \varepsilon$$

and $d_H(\mathcal{A}_\varepsilon, \mathcal{A}) \leq \varepsilon$ as required. We can express these observations in the the form of a theorem.

Theorem 1 *Let $\mathcal{F} = \{F_1, \dots, F_N\}$ be an IFS with strict contractions and attractor \mathcal{A} . Let Σ be the set of finite index sequences (k_1, \dots, k_m) , $k_i \in \{1, \dots, N\}$, satisfying the inequality*

$$\rho(F_{k_1} \cdots F_{k_m}) \leq \frac{\varepsilon}{\text{diam } \mathcal{A}} \leq \rho(F_{k_1} \cdots F_{k_{m-1}}).$$

For any point $x_0 \in \mathcal{A}$ the Hausdorff distance between the attractor and the set

$$\mathcal{A}_\varepsilon = \{x : x = F_{k_1} \cdots F_{k_m}(x_0), (k_1, \dots, k_m) \in \Sigma\}$$

is bound by ε ,

$$d_H(\mathcal{A}_\varepsilon, \mathcal{A}) \leq \varepsilon.$$

A complete recursive algorithm for approximating the attractor with a given accuracy is given below.

Example. In [2], Barnsley presented an iterated function system \mathcal{F} that defines a fern leaf (see the Appendix for a listing of the four transformations F_1, \dots, F_4). Figure 6 compares three different methods applied to render the attractor of \mathcal{F} . Since the contraction ratios of the transformations involved are significantly different from each other ($\rho(F_1) = 0.16$,

Algorithm	Adaptive-cut ($\mathcal{F}, R, z_0, \varepsilon$)
Purpose	Compute an image \mathcal{A}_ε of \mathcal{A} for an IFS \mathcal{F} with Hausdorff distance $d_H(\mathcal{A}_\varepsilon, \mathcal{A}) \leq \varepsilon$
Input	\mathcal{F} an IFS $\{F_1, \dots, F_N\}$
	R the diameter of the attractor
	z_0 a starting point in \mathcal{A}
	ε tolerance for Hausdorff distance
Global symbols	N the number of transformations
Local symbols	Id the identity transformation
Functions	$\rho()$ returns the contraction ratio of an affine map
Output	\mathcal{A}_ε the image of the attractor
begin	
	Descend (Id)
end	
Procedure	Descend (G)
Purpose	Plot $G(\mathcal{A})$, a portion of the attractor
Input	G composite affine transformation
Local symbols	k integer
Output	an image of $G(\mathcal{A})$
begin	
	if ($\rho(G)R \leq \varepsilon$) then
	Plot-Point ($G(z_0)$)
	else
	for $k = 1, \dots, N$ do
	Descend (GF_k)
	end for
	end if
end	

while $\rho(F_4) \approx 0.85$), the evaluation of the transformation tree to a certain depth yields a highly uneven distribution of points in the attractor (Figure 6b). The adaptive cut algorithm removes this drawback (Figure 6c) and provides more detail than the stochastic algorithm (Figure 6a).

The idea of the adaptive-cut algorithm can serve as a basis for a magnification algorithm, i.e. to render images of arbitrarily small details of the attractor, say in an area of interest \mathcal{R} . When in the course of the adaptive-cut algorithm a composite transformation $F_{k_1} \cdots F_{k_n}$ maps the attractor \mathcal{A} onto a subset $\mathcal{B} = F_{k_1} \cdots F_{k_n}(\mathcal{A})$ that is disjoint with the region of interest \mathcal{R} , $\mathcal{B} \cap \mathcal{R} = \emptyset$, then the entire subtree of further subdivisions of \mathcal{B} can be discarded. In this way the method automatically and quickly focuses on the region of interest \mathcal{R} , avoiding computations elsewhere. A related magnification algorithm has been proposed by Reuter in [16].

One question related to the adaptive cut algorithm is the use of the criterion

$$\rho(G)R \leq \varepsilon$$

to terminate the recursive descent of the transformation tree. This criterion is obviously sufficient to guarantee that the prescribed accuracy $d_H(\mathcal{A}_\varepsilon, \mathcal{A}) \leq \varepsilon$ is met. But is this condition necessary? In other words, would it be possible to stop descending sooner and still achieve the desired accuracy? It can be seen in iterated function systems without



Figure 6: A comparison of three attracting methods for the rendering of the set generated by an IFS. (a) The stochastic method using $N_1 = 198,541$ points. The probabilities used in the method are 0.01, 0.07, 0.07 and 0.85 corresponding to the four transformations given in the Appendix. (b) Evaluation of the tree of transformations to a given depth $n = 9$, resulting in $N_2 = 349,525$ points (often overlapping on the grid). (c) The adaptive-cut algorithm using $N_3 = N_1$ points.

overlap that, in general, there is no such short cut. For example, consider the Cantor set, defined by the following IFS transformations:

$$F_1(z) = \frac{z}{3}$$

$$F_2(z) = \frac{z}{3} + \frac{2}{3}$$

Here $\rho_1 = \rho_2 = \frac{1}{3}$ and the size of each portion of the attractor reached in the k -th stage of recursion is $\frac{1}{3^k}$. If such a subset does not contribute a point to the approximation \mathcal{A}_ε , where $\varepsilon = \frac{1}{3^k}$, then there are points y in \mathcal{A} such that

$$\min_{x \in \mathcal{A}_\varepsilon} \|x - y\| > \varepsilon.$$

Thus, in order to assure $d_H(\mathcal{A}, \mathcal{A}_\varepsilon) \leq \varepsilon$, it is clearly necessary that each subset of \mathcal{A} of size ε contributes a point to \mathcal{A}_ε . This demonstrates that the criterion used in the above algorithm to decide when to stop descending in the tree of images cannot be relaxed in general.

The attractive methods can be easily extended to three dimensions. For example, Plate 1 shows a 3D extension of the Sierpiński gasket (called a *fractal skewed web* in [11, page 142]), defined by an IFS with four transformations (see the Appendix). Points were represented as small spheres, and the resulting collection of spheres was ray-traced.

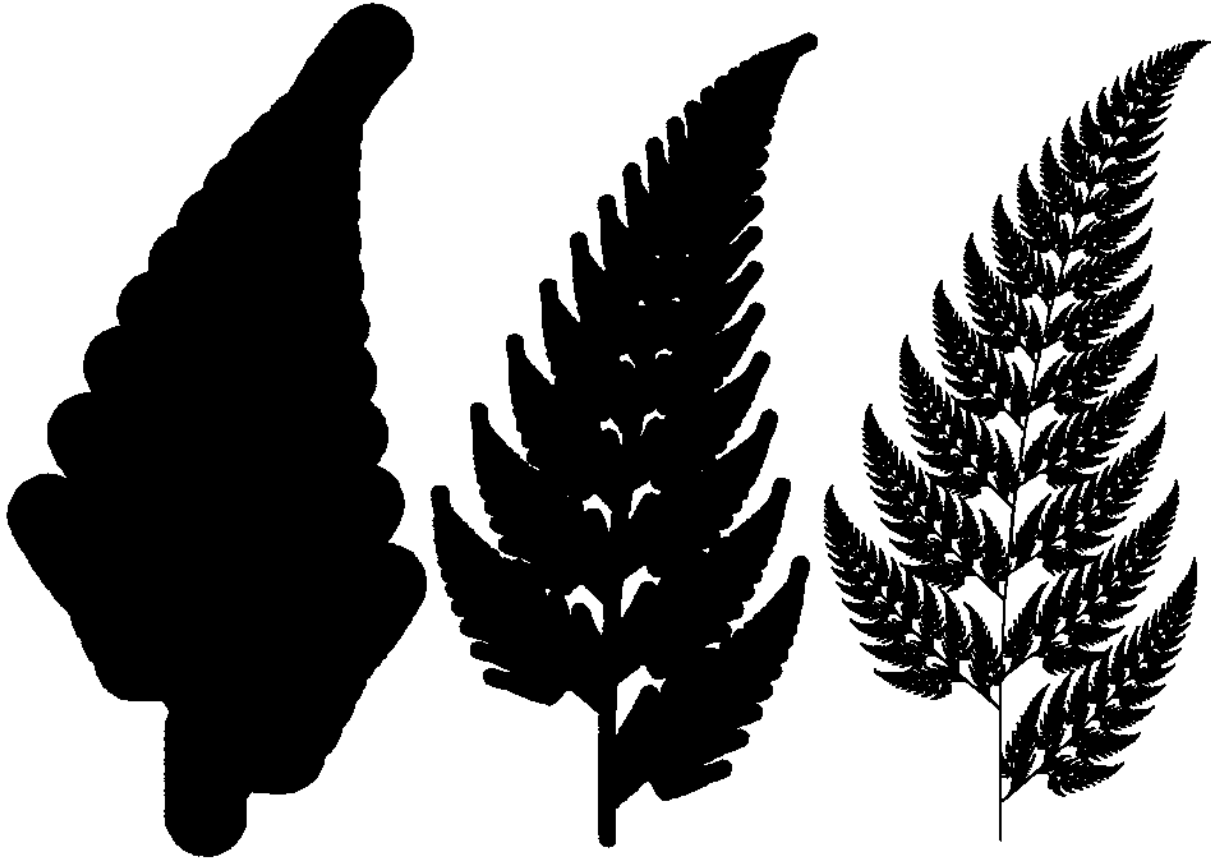


Figure 7: The adaptive cut algorithm may produce renderings of different resolution depending on the choice of the tolerance ε for the Hausdorff distance. Three different values are used in the above plots $\varepsilon = 0.5, 0.1, 0.015$. Each point was drawn as a small disk with appropriate radius such that the attractor is guaranteed to be covered by the image.

For practical purposes the attractive methods can be accelerated using the fact that the attractor can be displayed only with a finite screen resolution. The idea is to eliminate from further consideration images of more than one point falling in the same pixel [5]. This is of particular importance in the case of overlapping IFS's, since the adaptive cut algorithm ignores overlapping.

2.2 Estimating contraction factors

In this section we describe three methods to estimate the contraction ratio of composite affine transformations. The methods vary in ease of computation and in quality of the estimates.

The first and simplest method relies on the property $\rho(FG) \leq \rho(F)\rho(G)$ and estimates the contraction factor of a composite affine transformation $F_{k_1} \cdots F_{k_n}$ as the product of the individual contraction factors. Thus, $\rho(F_{k_1} \cdots F_{k_n}) \leq \rho(F_{k_1}) \cdots \rho(F_{k_n})$. This formula may grossly overestimate the actual value of the contraction ratio of the composite transformation. For example, consider the following two affine mappings:

$$F_1(z) = \frac{1}{100}Re(z) + \frac{99}{100}Im(z)i$$

$$F_2(z) = \frac{99}{100} \operatorname{Re}(z) + \frac{1}{100} \operatorname{Im}(z)i$$

Since $\rho(F_1) = \rho(F_2) = 99/100$, we obtain $\rho(F_1)\rho(F_2) = 99^2/100^2$. On the other hand, $\rho(F_1F_2) = 99/100^2$. Thus, the use of the product $\rho(F_1)\rho(F_2)$ overestimates the actual value of $\rho(F_1F_2)$ by the factor of 99.

We can use an alternative method for estimating the contraction ratio using the following property [9, page 57] that the contraction ratio $\rho(F)$ of an affine transformation $F : \mathbf{R}^n \rightarrow \mathbf{R}^n$, $F(x) = Ax + B$ with $A = (a_{ij})_{i,j=1,\dots,n}$ satisfies the inequality

$$\rho(F) \leq n \max_{i,j=1,\dots,n} |a_{ij}|.$$

In the above example this result provides a bound $2 \cdot 99/100^2$ which is much improved over $99^2/100^2$ but still off by a factor of 2. Also, note, that the estimate is not useful when $n \max_{i,j=1,\dots,n} |a_{ij}| > 1$.

The third method is more expensive computationally than the previous two, but provides exact values of the contraction factors. It uses the fact, that the contraction ratio $\rho(F)$ of the affine transformation $F(x) = Ax + B$, can be expressed as the square root of the maximal eigenvalue of $A^T A$ [17, section 4.4]:

$$\rho(F) = \sqrt{\max\{|\lambda_i| : \lambda_i \text{ eigenvalue of } A^T A\}}.$$

In the two-dimensional case ($n = 2$) the evaluation of $\rho(F)$ involves the computation of two square roots. As one is interested not in the exact contraction ratio but only in a tight bound of it, one may replace the square root computation by a properly organized table look-up procedure, which will speed up the method considerably.

In the above discussion contraction ratios are defined relative to the Euclidean metric in \mathbf{R}^n . However, the necessary eigenvalue computation may be costly, especially for larger dimensions n . Thus, it may be of advantage to switch to a different metric, given by the maximum norm

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i|.$$

In this case the contraction ratio can be computed efficiently by the formula

$$\rho_\infty(F) = \max_{i=1,\dots,n} \sum_{j=1}^n |a_{ij}|,$$

where $F(x) = Ax + B$ with $A = (a_{ij})$ as above [9, page 57].

2.3 Extension of the adaptive-cut algorithm to render the invariant measure

A small modification allows the adaptive-cut method to render the invariant measure in addition to the geometry of the attractor \mathcal{A} , which will still be approximated with prescribed accuracy. Let p_1, \dots, p_N denote the probabilities for the transformations $F_1, \dots, F_N \in \mathcal{F}$. The invariant measure μ is the fixed point of the Markov operator M :

$$\mu = M(\mu) = \sum_{i=1}^N p_i \mu F_i^{-1}.$$

Iterating this equation we obtain

$$\mu = \sum_{i_1, \dots, i_n=1}^N p_{i_1} \cdots p_{i_n} \mu_{F_{i_n}^{-1} \cdots F_{i_1}^{-1}} .$$

The transformed measures $\mu_{F_{i_n}^{-1} \cdots F_{i_1}^{-1}}$ have unit mass and support $F_{i_1} \cdots F_{i_n}(\mathcal{A})$. In the adaptive-cut method the sets are rendered as points, provided their diameters are sufficiently small to meet the prescribed tolerance $\varepsilon > 0$. To compute the measure μ , we accumulate the weight factors $p_{i_1} \cdots p_{i_n}$ in the pixels corresponding to the points being drawn. Thus, at the termination of the algorithm, pixels with positive accumulated weights approximate the attractor \mathcal{A} in the same fashion as provided by the basic adaptive-cut method while a rendering of the accumulated weights will display the invariant measure at the resolution of the image.



Figure 8: Rendering of the fern using the stochastic method with improved probabilities 0.03, 0.11, 0.13, 0.73. The same number of points as in Figure 6(a) has been used.

2.4 Estimating probabilities for the chaos game

The stochastic method requires a set of probabilities $p_k, k = 1, \dots, N$ that determine which of the transformations F_1, \dots, F_N should be taken in each step of the algorithm. The choice of these probabilities is not obvious. It is suggested in [2, page 87] to set

$$p_k \approx \frac{|\det A_k|}{\sum_{i=1}^N |\det A_i|}$$

where $A_k, k = 1, \dots, N$ denotes the linear part of the corresponding transformation. If one of the determinants is zero, a small positive probability should be used. The adaptive-cut method may provide a better way to assign values for the probabilities. We subdivide the

points plotted by the adaptive cut method into N subsets, each one collecting points drawn in the corresponding set $F_k(\mathcal{A})$. The relative number of points in each subset determines the corresponding probability. For example, for the fern we obtain the numbers 0.03, 0.11, 0.13, 0.73. When used as probabilities in the chaos game, these values result in an image with more evenly distributed points as compared to images based on the chaos game using probabilities suggested by the above formula (compare Figure 8 with Figure 6). However, the probabilities as listed above should not be taken as absolute because their values depend to some extent on the resolution of the image. Other weight factors may be better for other resolutions.

2.5 Ray tracing fractals generated by IFS's

The methods for generating a point set \mathcal{A}_ϵ approximating the attractor \mathcal{A} also apply to IFS's operating in three (and more) dimensions. However, the process of visualizing the resulting data sets is more difficult. For images with a highly realistic touch, ray tracing is often the method of choice [8]. The basic operation in the ray tracing scheme is the computation of the first intersection of a given ray in a three-dimensional scene with the set of all objects contained in the scene. In the case of an IFS attractor \mathcal{A} we represent the approximating point set \mathcal{A}_ϵ as a set of small spheres covering the attractor. Spheres are very easy to ray trace, because the ray surface intersection calculations are simple and fast as compared with other surfaces. Moreover, the use of hierarchical bounding volumes [8], one of the standard methods for accelerating the ray tracing process, lends itself very naturally to attractors of IFS's.

As in the adaptive cut method the attractor \mathcal{A} of an IFS $\{F_1, \dots, F_N\}$ is represented as the union of its images $F_k(\mathcal{A})$, $k = 1, \dots, N$. Each set $F_k(\mathcal{A})$ is contained in a sphere which serves as a bounding volume. Only if the ray pierces this sphere, the set $F_k(\mathcal{A})$ is subdivided into smaller sets $F_k F_i(\mathcal{A})$, $i = 1, \dots, N$, each one contained in a proper bounding sphere. This subdivision is recursively continued until the final resolution is achieved and the intersection of the ray with the sphere closest to the ray origin is rendered.

An additional saving in compute time is possible by constructing the tree of bounding volumes in such a manner that always the bounding volume closest to the ray origin is searched first for a ray surface intersection. In this approach, bounding spheres further away often may be discarded, even when the ray intersects the volume.

Plate 1 shows an example of a ray traced three-dimensional attractor.

3 Rendering methods based on the Euclidean distance from the fractal

3.1 Distance estimation using a fixed approximation of the attractor

The distance of a point x to the attractor \mathcal{A} is defined by the formula

$$d(x, \mathcal{A}) = \inf_{y \in \mathcal{A}} \|x - y\| .$$

In practice, this formula cannot be directly applied to calculate the distance d , since \mathcal{A} may contain an infinite number of points. However, we can effectively estimate d using a finite point set $\mathcal{A}_\epsilon \subset \mathcal{A}$, approximating \mathcal{A} with a given accuracy $d_H(\mathcal{A}, \mathcal{A}_\epsilon) \leq \epsilon$:

$$|d(x, \mathcal{A}) - d(x, \mathcal{A}_\epsilon)| \leq \epsilon . \quad (2)$$

The adaptive-cut method readily yields such an approximation \mathcal{A}_ϵ . Distance computation based on a finite approximation of \mathcal{A} is easy to implement. However, the time needed to compute the distance d from a given sample point x to the set \mathcal{A}_ϵ is directly proportional to the number of points y in \mathcal{A}_ϵ . For a high-resolution approximation of \mathcal{A} , this number is of the order 10^5 or 10^6 , resulting in long rendering times. Below we present a method which allows for faster computation of distance d if high-resolution rendering is needed.

3.2 Distance calculation using region subdivision method

We may cut down on the number of points in the attractor, necessary to compute the distance $d(x, \mathcal{A})$, using the following idea. The attractor \mathcal{A} of an IFS $\mathcal{F} = \{F_1, \dots, F_N\}$ satisfies the equality:

$$\mathcal{A} = \bigcup_{i=1}^N F_i(\mathcal{A}) .$$

Thus, while computing the distance $d(x, \mathcal{A})$, we can discard *all* points in $F_i(\mathcal{A})$, once we have found a point y in some other $F_j(\mathcal{A})$ closer to x than the set $F_i(\mathcal{A})$ (Figure 9).

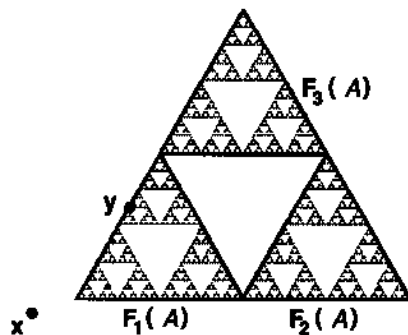


Figure 9: In this Sierpiński triangle a point $y \in F_1(\mathcal{A})$ is closer to the sampling point x than the sets $F_2(\mathcal{A})$ and $F_3(\mathcal{A})$. Therefore in the search for the minimal distance to x , all points from $F_2(\mathcal{A})$ and $F_3(\mathcal{A})$ can be discarded.

We do not know the sets $F_i(\mathcal{A})$ but we can make an estimate. If R is the diameter of \mathcal{A} and $\rho_i < 1$ is the contraction factor of $F_i \in \mathcal{F}$, then $F_i(\mathcal{A})$ is contained in a disk of radius $\rho_i R$ about $F_i(y_0)$, where y_0 is an arbitrary point in \mathcal{A} . Formally,

$$F_i(\mathcal{A}) \subset D_{\rho_i R}(F_i(y_0)) .$$

Thus, we will discard $F_i(\mathcal{A})$, if its bounding disk is farther away from x than some known point in \mathcal{A} (Figure 10). This idea is carried out iteratively, by subdividing the regions remaining after the previous step. The recursive subdivision is continued until the minimum distance $d(x, \mathcal{A})$ is reached within a specified tolerance. The composite transformations must be *premultiplied* so that, given a composite transformation $G =$

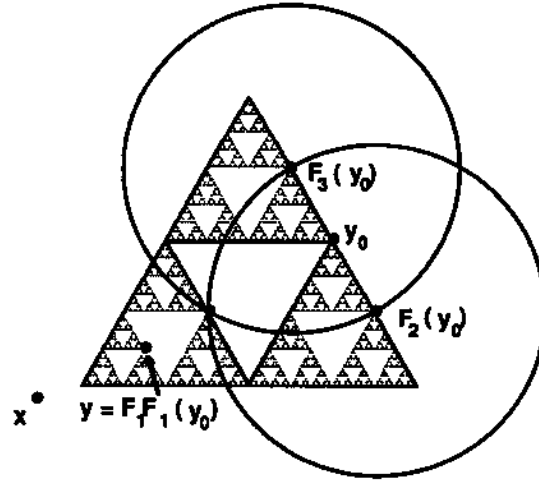


Figure 10: When computing the distance of x to \mathcal{A} we may discard all points in $F_2(\mathcal{A})$ and $F_3(\mathcal{A})$, because a known point $y = F_1(F_1(y_0))$ is closer to x than the disks that include $F_2(\mathcal{A})$ and $F_3(\mathcal{A})$.

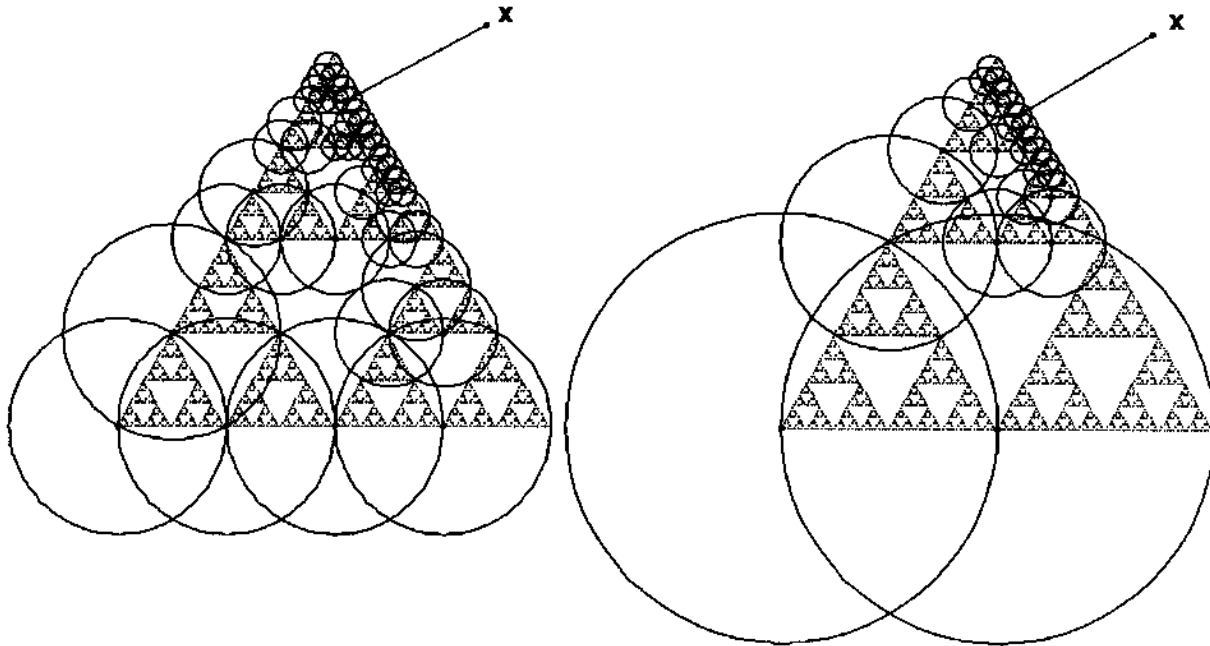


Figure 11: Computation of the distance to a fractal by the adaptive subdivision algorithm: (a) coherence is not employed, (b) the same sampling point x is selected for the second time, coherence is employed.

$F_{i_1} F_{i_2} \cdots F_{i_n}$, the subsequent mappings $G F_{i_{n+1}}, i_{n+1} = 1, \dots, N$ properly subdivide $G(\mathcal{A})$ into smaller regions. Specifically, once the set $G(\mathcal{A})$ is removed from further consideration, no parts of $G(\mathcal{A})$ will be unnecessarily evaluated (in systems without overlap). See the algorithm `adaptive-subdivision` for details.

3.3 The use of point-to-point coherence

Let $y_{min}(x)$ denote a point in \mathcal{A} closest to x with a given tolerance ε . Thus, $d(x, y_{min}(x)) \approx d(x, \mathcal{A})$. Although y_{min} is not a continuous function of x , the distance $d(x, \mathcal{A})$ is continuous in x . Thus, if x' is a new sampling point lying in the vicinity of x , the value $d(x', y_{min}(x))$

Algorithm	Adaptive-subdivision ($\mathcal{F}, R, y_0, x, \epsilon$)
Purpose	Estimate the distance of x to the attractor \mathcal{A} of an IFS \mathcal{F} with accuracy ϵ
Input	\mathcal{F} an IFS $\{F_1, \dots, F_N\}$ R the diameter of the attractor y_0 a starting point in the attractor x a sampling point ϵ tolerance for distance estimation
Global symbols	N the number of transformations
Local symbols	Id the identity transformation $List$ a list of 3-tuples, (G_k, r_k, d_k) , where: G_k is a composed affine transformation r_k is an estimate of the diameter of $G_k(\mathcal{A})$ d_k is the distance between x and $y_k = G_k(y_0) \in G_k(\mathcal{A})$ d_{min} smallest known distance between x and a point in \mathcal{A} y_{min} a point in \mathcal{A} such that $d(x, y_{min}) = d_{min}$ k integer
Output	d_{min} distance estimate such that $ d_{min} - d(x, \mathcal{A}) \leq \epsilon$ y_{min} a point in \mathcal{A} such that $d(x, y_{min}) = d_{min}$

begin

```

/* initialization */
ymin = y0
dmin = d(x, ymin)
List = {(Id, R, dmin)}
do
  /* Create a new List containing information about the subsets of A */
  List = Process-list(List)
  /* Find the element of List closest to x */
  for k = 1, ..., length(List) do
    if (dk < dmin) then
      ymin = Gk(y0)
      dmin = dk
    end if
  end for
  /* Eliminate item k from List if the diameter of Gk(A) is smaller than epsilon (resolution
of the algorithm) or if the point ymin is closer to x than the set Gk(A) */
  for k = 1, ..., length(List) do
    if (rk < epsilon) or (dmin <= dk - rk) then
      mark k-th element from the List for deletion
    end if
  end for
  delete all marked elements from the List
until (List is empty, i.e. length(List) = 0)
return (dmin, ymin)
end

```

provides a good estimate of the distance $d(x', \mathcal{A})$. We can use $y_{min}(x)$ and $d(x', y_{min}(x))$ as the initial point y_{min} and distance d_{min} set during the initialization process of the **adaptive-subdivision** algorithm. As a result, the value of d_{min} is approximated well from the beginning of the algorithm, and some regions can be rejected in earlier steps than if an arbitrary starting point y_0 were used, resulting in increased algorithm speed. This is illustrated in Figure 11. The circles enclose regions being discarded. Relatively

Procedure **Process-list** ($List$)
Purpose Create a new $List$ containing information about the subsets of A
Input $List$ a list of n 3-tuples (G_k, r_k, d_k)
Local symbols i, k, l integers
Functions $\rho()$ returns estimate for contraction ratio
Output \widehat{List} output list of 3-tuples $(\widehat{G}_k, \widehat{r}_k, \widehat{d}_k)$
begin
 $l = 1$
 for $k = 1, \dots, \text{length}(List)$ **do**
 for $i = 1, \dots, N$ **do**
 $\widehat{G}_l = G_k F_i$
 $\widehat{r}_l = \rho(\widehat{G}_l) R$
 $\widehat{d}_l = d(x, \widehat{G}_l(y_0))$
 $l = l + 1$
 end for
 end for
 return (\widehat{List})
end

ε	method	dragon	fern	gasket	star
0.05	fixed approx.	1.95	4.88	0.94	2.75
	adp. subdiv.	21.84	125.33	5.28	33.27
	with coher.	18.26	108.96	3.88	28.97
0.01	fixed approx.	97.63	163.03	9.75	103.70
	adp. subdiv.	48.25	564.93	10.81	72.43
	with coher.	41.23	355.31	8.16	54.22
0.005	fixed approx.	392.81	707.05	40.75	408.44
	adp. subdiv.	62.24	826.22	15.93	82.90
	with coher.	50.45	483.30	11.05	62.05

Table 1: CPU time (in minutes) for the computation of the distance function for three different iterated function systems in a grid of 256×256 points. The experiment was performed on a MIPS M-120 computer using the R-2000 processor. The transformations that define the four IFS's are given in the Appendix.

larger regions are discarded if the point-to-point coherence is employed. For example, Table 1 compares total computation times from several sample sets, calculated in a grid of 256×256 points. The grid was scanned in a spiral way, from the window borders towards the center.

Three different values of the tolerance for distance estimation ε are considered. The data obtained using the adaptive subdivision method for $\varepsilon = 0.005$ were used to generate Plate 2. Table 1 shows that, for small values of tolerance ε , the adaptive subdivision algorithm is faster than the fixed approximation method.

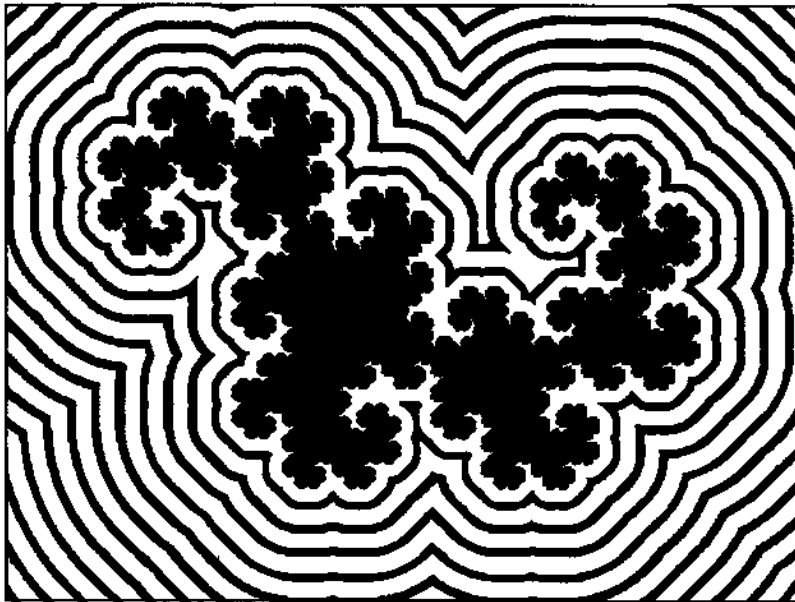
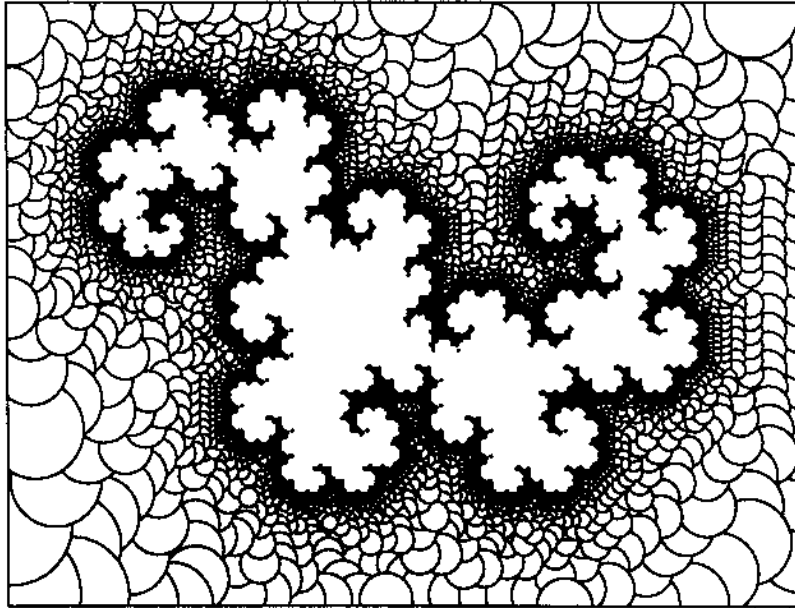


Figure 12: Two visualizations of the distance function for the dragon curve.

3.4 Visualization of the distance function

The simplest method for visualizing the distance function computed for an array of points associates different colors to separate intervals of distance values. This technique was applied to create Plate 3 and Figure 12 (bottom). Instead of varying colors, one can vary the height of the points, thus representing the distance function as a height field (Plates 2 and 4). Another possibility is to draw disks with the radius r proportional to the distance from the disk center to the fractal. Once a disk is drawn, all points inside it are removed from further consideration (they will lie outside of the fractal), and the center of the next disk is chosen on the boundary of some previously drawn disk. This is an extension of the technique introduced by Fisher [13, pages 287-296] to generate images of Mandelbrot

and Julia sets. An example is shown in Figure 12 (top). The radius of each circle has been chosen at pleasure to be equal to $\frac{1}{4}$ of the distance between the circle center and the fractal. The order of drawing is indicated by hidden line elimination (new disks are drawn “on top” of the previous ones). A similar approach was applied to create Plates 5 and 6. The circles were replaced by spheres, and the resulting scene was ray traced.

4 Rendering methods based on the escape time

In order to define the escape-time function for iterated function systems, let us review the analogous concept for Julia sets. The Julia set for the quadratic mapping $z \mapsto z^2 + c$ can be viewed as the attractor \mathcal{A} defined by the *nonlinear* IFS $\{F_1(z) = \sqrt{z - c}, F_2(z) = -\sqrt{z - c}\}$. Given a point z not in the filled-in Julia set, repeated application of the inverse mappings F_1^{-1} and F_2^{-1} will eventually take the initial point z to ∞ . The number of iterations necessary to force the point outside of a large disk of radius R is the escape time of z . This number is *independent* of the particular sequence of the applied inverses F_1^{-1}, F_2^{-1} because they collapse into the same map $R_c : z \mapsto z^2 + c$. A continuous — and even differentiable — extension of the integer-valued escape time function $E_c(z)$ is given by means of the *potential function* [4,13]:

$$g_c(z) = \lim_{k \rightarrow \infty} \frac{\log |R_c^k(z)|}{2^k} \quad (3)$$

and the formula

$$E_c(z) = -\log_2 g_c(z) .$$

Here $R_c(z) = z^2 + c$ and $R_c^k(z)$ denotes the k -fold iteration of R_c applied to the point z .

4.1 Discrete escape time for IFS's

The concept of the escape-time function can be extended to iterated function systems in which all transformations are invertible. In [15] it was shown that given an IFS $\mathcal{F} = \{F_1, \dots, F_n\}$, for any point x of the attractor \mathcal{A} there exists at least one infinite sequence of inverse mappings $F_{i_1}^{-1}, F_{i_2}^{-1}, F_{i_3}^{-1}, \dots$ which will **not** take x to ∞ . In contrast, any sequence of inverse mappings applied to a point $x \notin \mathcal{A}$ will eventually repel x to infinity. For a point x outside of the attractor we may therefore define the integer-valued escape-time function as the maximal number of inverse iterations necessary to map x outside of a large disk of radius R :

Definition 2 Let $\mathcal{F} = \{F_1, \dots, F_n\}$ be an IFS with invertible transformations and D_R be the disk of radius R centered at zero. Furthermore, let $R > 0$ be a number sufficiently large to satisfy the condition

$$F_k(D_R) \subset D_R, \quad k = 1, \dots, N . \quad (4)$$

The (integer-valued) escape-time function is defined as follows

$$E_R : X \setminus \mathcal{A} \rightarrow \{0, 1, 2, \dots\}$$

$$x \mapsto \begin{cases} 0 & \text{if } \|x\| \geq R \\ 1 + \max_{k=1, \dots, N} E_R(F_k^{-1}(x)) & \text{otherwise} \end{cases} \quad (5)$$

The condition on R ensures that a point, once mapped outside of the disk of radius R , cannot return to the disk when subsequent inverse transformations are applied. Such R always exists since we assume that all transformations are strictly contractive.

Besides the nonlinearity, the fundamental difference between the mappings defining Julia sets and IFS's is that the inverses of the affine maps do not collapse into a single map. Consequently, the order of applications of inverse transformations involved in the calculation of the escape time is important. In principle, the appropriate sequence of transformations can be determined by dividing the entire plane (case $X = \mathbf{R}^2$) into N disjoint regions. The transformation applied to a point is determined by the region this point is in. This method for calculating the escape-time function was introduced independently in [2, chapter 7] and [14,15]. The key difficulty lies in the definition of domain boundaries which will yield the longest sequences of transformations according to formula (5). Specifically, this boundary may be a fractal curve itself. A more general "brute force" method proposed in [15] is to recursively construct the entire tree of points which:

- result from the application of a sequence of inverse transformations $F_{i_1}^{-1}, F_{i_2}^{-1}, \dots, F_{i_n}^{-1}$ to the sampling point x , and
- remain within the large disk of radius R .

The length of the longest path in this tree is the escape-time value $E_R(x)$.

As mentioned above, the sequences of inverse transformations can be associated with a subdivision of the complement of the attractor. This subdivision is formalized using the notion of *index maps*

$$I_k : D_R \setminus \mathcal{A} \rightarrow \{1, \dots, N\}$$

$$y \mapsto i_k,$$

where the index i_k is taken from the sequence i_1, i_2, \dots chosen such that F_{i_1}, F_{i_2}, \dots determines the escape time of y . For example, the set of points with index $I_1 = 2$, noted $I_1^{-1}(2)$, is the region in which the longest sequence of preimages that remain in the disk is such that the first inverse map that is used in it is F_2^{-1} . This region again can be subdivided according to I_2 : $I_1^{-1}(2) \cap I_2^{-1}(3)$ is the set of initial points whose sequence of preimages is generated first with an application of F_2^{-1} and then one of F_3^{-1} , etc. Plate 7 shows these regions for the Sierpiński gasket.

Below we introduce a generalization of the escape-time function that extends its range to real numbers in such a way that $E_R(x)$ is a continuous function of the variable x .

4.2 Continuous extension of the escape-time function

Definition 3 Let $\mathcal{F} = \{F_1, \dots, F_n\}$ be an IFS with invertible transformations and $R > 0$ satisfy condition (4). The (real-valued) escape-time function is defined as follows

$$E_R : X \setminus \mathcal{A} \rightarrow \mathbf{R}$$

$$x \mapsto \begin{cases} 0 & \text{if } \|x\| \geq R \\ \max_{k=1, \dots, N} \frac{\log R - \log \|x\|}{\log \|F_k^{-1}(x)\| - \log \|x\|} & \text{if } \|x\| < R, \text{ and} \\ 1 + \max_{k=1, \dots, N} E_R(F_k^{-1}(x)) & \text{if } \|F_k^{-1}(x)\| \geq R, k = 1, \dots, N \\ & \text{otherwise} \end{cases} \quad (6)$$

It is easy to see the origin of this definition using the (trivial) IFS consisting of only one transformation $F(x) = rx$, $0 < r < 1$, as an example. For the integer-valued escape-time function we have

$$E_R(x) = k \text{ if } r^k R \leq \|x\| < r^{k-1} R .$$

Taking logarithms,

$$E_R(x) - 1 < \frac{\log \|x\|/R}{\log r} \leq E_R(x) .$$

Thus, $E_R(x)$ is just the next largest integer to $\log \frac{\|x\|}{R} / \log r$. This expression is equal to

$$\frac{\log \frac{\|x\|}{R}}{\log r} = \frac{\log \|x\| - \log R}{\log \|x\| - \log \frac{\|x\|}{r}} = \frac{\log R - \log \|x\|}{\log \|F^{-1}(x)\| - \log \|x\|}$$

and serves as a continuous, and in this case even differentiable, extension of the integer-valued escape-time function. The definition above generalizes this concept to arbitrary IFS's.

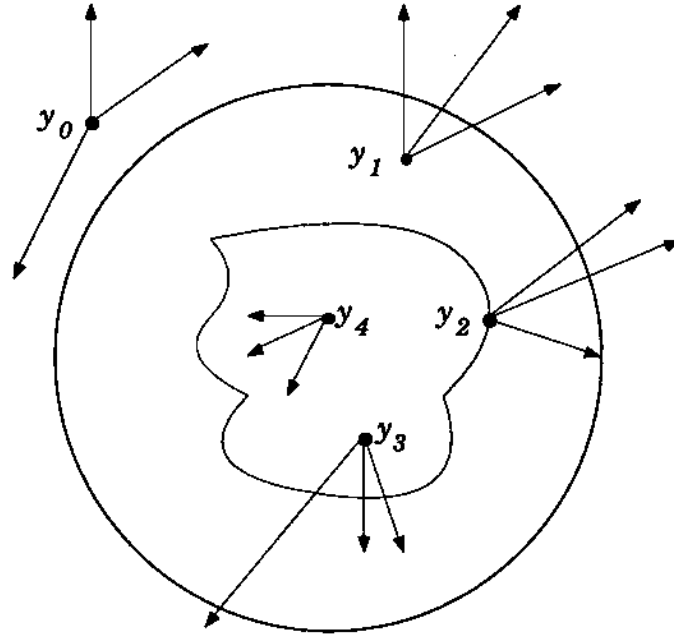


Figure 13: In this schematic diagram five points y_0, \dots, y_4 are shown with their preimages with respect to all transformations of an IFS $\{F_1, F_2, F_3\}$. The point y_0 satisfies the inequality $\|y_0\| \geq R$, thus $E_R(y_0) = 0$. Furthermore, $\|y_1\| < R$, but all preimages of y_1 have norm greater than R , thus, $0 < E_R(y_1) < 1$. Point y_2 has two preimages outside of the disk and one on the boundary of the disk, hence $E_R(y_2) = 1$. The closed curve, of which y_2 is a point, collects all points with escape time equal to 1. Hence, $1 < E_R(y_3) < 2$. For y_4 we have $E_R(y_4) > 2$.

Theorem 4 *The escape-time function $E_R(x)$ depends continuously on the point x and on the parameter R .*

Proof — proceeds by examining cases shown in Figure 13. We present the proof only for the dependence on the point x , because the proof for the continuity with respect to

R is similar. Outside the disk D_R the escape-time function $E_R(x)$ is equal to 0, thus it is continuous. On the boundary of the disk continuity holds since the expression

$$\max_{k=1,\dots,N} \frac{\log R - \log \|x\|}{\log \|F_k^{-1}(x)\| - \log \|x\|} \quad (7)$$

is also equal to 0 for $\|x\| = R$. If $\|x\| < R$ and $\|F_k^{-1}(x)\|$ is strictly greater than R for all $k = 1, \dots, N$, then $E_R(x)$ is equal to the expression (7) in a neighborhood of x , therefore also continuous. Due to the recursive nature of the definition of $E_R(x)$ it remains to check that $E_R(x)$ is continuous at points x with $\|F_k^{-1}(x)\| \geq R$ for all $k = 1, \dots, N$, and where equality holds for at least one index k (see point y_2 in Figure 13). Consider a sequence of points x_i , $i = 1, 2, \dots$ such that $x_i \rightarrow x$ as $i \rightarrow \infty$. From the definition of the escape time we have $E_R(x) = 1$ while for sufficiently large indices i we have that $E_R(x_i)$ is equal to either

$$\max_{k=1,\dots,N} \frac{\log R - \log \|x_i\|}{\log \|F_k^{-1}(x_i)\| - \log \|x_i\|} \quad (8)$$

or

$$1 + \max_{k=1,\dots,N} E_R(F_k^{-1}(x_i)) \quad (9)$$

depending on whether all preimages of x_i have norms greater or equal to R or not. Clearly, the first expression (8) tends to $E_R(x) = 1$ as $i \rightarrow \infty$. To investigate the second expression (9), we note that $\|F_k^{-1}(x)\| \geq R$, $k = 1, \dots, N$. Moreover, we already know that E_R is continuous at the points $F_k^{-1}(x)$ for $k = 1, \dots, N$. Thus, we conclude that the expression (9) must tend to

$$1 + \max_{k=1,\dots,N} E_R(F_k^{-1}(x))$$

as $i \rightarrow \infty$. The value of this limit of course is equal to 1. \square

The practical computation of the escape-time function may proceed directly from the definition by computing a tree of preimages of a point under all inverse transformations until all points escape the disk of radius R (see procedure `build-escape-time-tree` below). This may be a very time consuming calculation when it is done many times as in the computation of an image displaying escape times in a whole region. In Sections 4.4 and 4.5 we propose two schemes that potentially may speed up the computation.

Plates 8, 9, 10 and 11 display visualizations of the escape-time function for the dragon curve, the Sierpiński gasket and the fern. Shown in each image is the surface given by the graph of the escape-time function or a top view with color coded values of the function. Plate 12 compares the escape-time and distance function for a Cantor set.

4.3 Weighed average escape-time function

The escape time $E_R(x)$ is defined by the longest sequence of preimages of x remaining in the disk of radius R . All other sequences of preimages have no effect on the escape time. Thus, only an extreme case is taken into account. In this section we present a modification of escape time in which all leaves of the tree of preimages of a point contribute to the point's escape time. For this *weighed average escape time* we associate weights $p_1, \dots, p_N > 0$, $\sum_{i=1}^N p_i = 1$, with the affine transformations F_1, \dots, F_N of an IFS. These weights naturally extend to the leaves of the tree of preimages of a point. As

shown in Figure 14, each leaf contributes a weighed term derived from the logarithmic expression given in Definition 3. The purpose of the weighed average escape-time function is to describe the expected behavior of the chaos game, when it is run *backwards*. All point sequences generated by iteration of the inverse transformations must diverge, provided they are started in the complement of the IFS attractor. The weighed average escape-time is the expected number of iterations for a point to escape from a disk of a given radius.

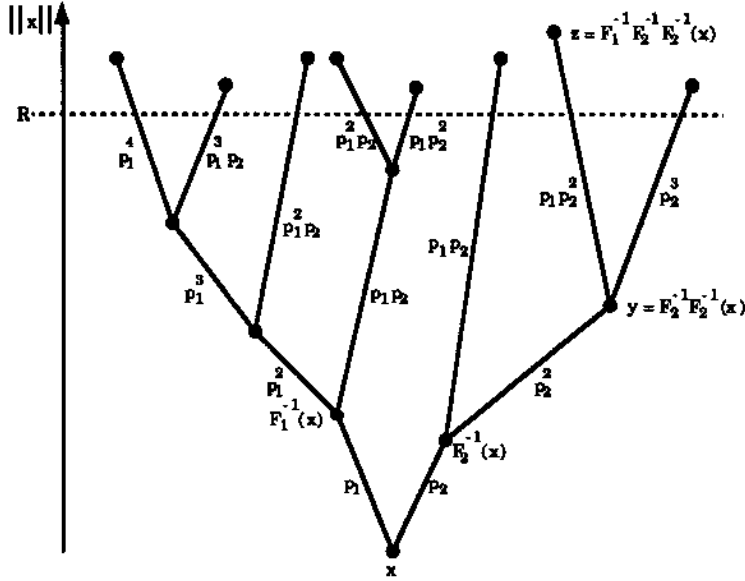


Figure 14: Tree of preimages for the weighed average escape time. Each point z with $\|z\| \geq R$ in the tree (a leaf point) yields a contribution to the weighed average escape time of the point x according to formula (10). For example, the point $z = F_1^{-1} F_2^{-1} F_2^{-1}(x) = F_1^{-1}(y)$ yields a term $p_1 p_2^2 \cdot (2 + (\log R - \log \|y\|) / (\log \|z\| - \log \|y\|))$.

Definition 5 Let $\mathcal{F} = \{F_1, \dots, F_N\}$ be an IFS with invertible transformations and $R > 0$ satisfy condition (4). The weighed average escape-time function $\bar{E}_R : X \setminus \mathcal{A} \rightarrow \mathbf{R}$ is defined as follows. Let $\mathcal{S}_R(x)$ be the set of finite index sequences (k_1, \dots, k_{m+1}) , $k_i \in \{1, \dots, N\}$ such that

$$\|F_{k_m}^{-1} \cdots F_{k_1}^{-1}(x)\| < R \quad \text{and} \quad \|F_{k_{m+1}}^{-1} \cdots F_{k_1}^{-1}(x)\| \geq R.$$

Then

$$\bar{E}_R(x) = \sum_{(k_1, \dots, k_{m+1}) \in \mathcal{S}_R(x)} p_{k_1} \cdots p_{k_{m+1}} \cdot \left(m + \frac{\log R - \log \|x_m\|}{\log \|x_{m+1}\| - \log \|x_m\|} \right) \quad (10)$$

where

$$x_m = F_{k_m}^{-1} \cdots F_{k_1}^{-1}(x), \quad x_{m+1} = F_{k_{m+1}}^{-1} \cdots F_{k_1}^{-1}(x).$$

Another description of $\bar{E}_R(x)$ that uses recursion is also given: When $\|x\| \geq R$ we have $\bar{E}_R(x) = 0$. In the other case

$$\bar{E}_R(x) = \sum_{k=1}^N p_k \cdot \begin{cases} 1 + \bar{E}_R(F_k^{-1}(x)) & \text{if } \|F_k^{-1}(x)\| < R \\ \frac{\log R - \log \|x\|}{\log \|F_k^{-1}(x)\| - \log \|x\|} & \text{otherwise} \end{cases} \quad (11)$$

holds, provided that $x \notin \mathcal{A}$.

The weighed average escape-time function $\bar{E}_R(x)$ is continuous like the real-valued escape-time function $E_R(x)$. We omit the proof, which is very similar to the proof of Theorem 4.

The real-valued escape-time function $E_R(x)$ tends to infinity as x approaches the attractor. This may also be true for the weighed average escape-time function, as the simple IFS consisting of only one transformation $F_1(x) = rx$, $|r| < 1$ shows ($\lim_{x \rightarrow 0} \bar{E}_R(x) = \infty$). However, it is not true in general. For this phenomenon we provide an example.

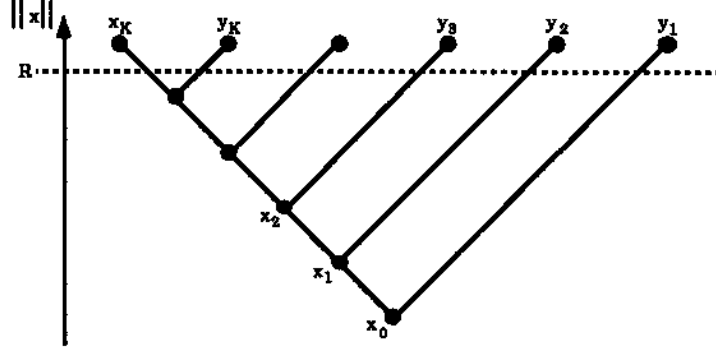


Figure 15: Tree of preimages for computation of the weighed average escape time at the initial point x_0 . The IFS is $\mathcal{F} = \{F_1, F_2\}$, with inverse transformations $F_1^{-1}(x) = 2x - 1$, $F_2^{-1}(x) = 2x + 1$ for $x \in \mathbb{R}$ and $p_1 = p_2 = \frac{1}{2}$. The initial point x_0 is greater than 1 by a small amount. The radius is $R = 2$.

Let $\mathcal{F} = \{F_1, F_2\}$, $F_1(x) = \frac{1}{2}x + \frac{1}{2}$, $F_2(x) = \frac{1}{2}x - \frac{1}{2}$ for real x , and $p_1 = p_2 = \frac{1}{2}$. The attractor is the real interval $[-1, 1]$. We choose $R = 2$ and compute $\bar{E}_R(x_0)$ as x_0 tends to 1 from above. Given an initial point $x_0 > 1$ we define points $x_k, y_k, k = 1, 2, \dots$ via $x_{k+1} = F_1^{-1}(x_k)$ and $y_{k+1} = F_2^{-1}(x_k)$, see Figure 15. We obtain

$$\begin{aligned} x_{k+1} &= F_1^{-1}(x_k) = 2x_k - 1 = 2^{k+1}(x_0 - 1) + 1, \\ y_{k+1} &= F_2^{-1}(x_k) = 2x_k + 1 = 2^{k+1}(x_0 - 1) + 3 > R = 2. \end{aligned}$$

We define the integer K by $x_{K-1} < R \leq x_K$. Thus, the leaf nodes in the tree of preimages (Figure 15) are given by x_K and y_1, \dots, y_K . The weighed average escape time for x_0 is

$$\bar{E}_R(x_0) = \frac{1}{2^K} \left(K - 1 + \frac{\log R - \log x_{K-1}}{\log x_K - \log x_{K-1}} \right) + \sum_{k=1}^K \frac{1}{2^k} \left(k - 1 + \frac{\log R - \log x_{k-1}}{\log y_k - \log x_{k-1}} \right).$$

As we let $x_0 \rightarrow 1$ we have $x_k \rightarrow 1$, $y_k \rightarrow 3$ and $K \rightarrow \infty$. Thus, in the limit the first term in $\bar{E}_R(x_0)$ vanishes (note that the logarithmic fraction is bounded by 1), and the other term yields

$$\lim_{x_0 \rightarrow 1^+} \bar{E}_R(x_0) = \sum_{k=1}^{\infty} \frac{1}{2^k} \left(k - 1 + \frac{\log 2}{\log 3} \right) = 1 + \frac{\log 2}{\log 3}.$$

In this example we may extend the definition of weighed average escape time to all points in \mathbb{R} including the attractor \mathcal{A} using the formulae (10) and (11). Assuming that $\bar{E}_R(1)$ is finite we obtain for $x = 1 \in \mathcal{A}$

$$\bar{E}_R(1) = \frac{1}{2} \left(\frac{\log 2}{\log 3} + 1 + \bar{E}_R(1) \right)$$

which, solved for $\bar{E}_R(1)$, yields

$$\bar{E}_R(1) = 1 + \frac{\log 2}{\log 3}$$

in agreement with the limit for $x \rightarrow 1^+$. In this IFS all weighed average escape times for points in the attractor appear to be finite, e.g.

$$\bar{E}_R(0) = \frac{1}{2}(2 + \bar{E}_R(1) + \bar{E}_R(-1)) = 2 + \frac{\log 2}{\log 3}.$$

4.4 Acceleration by the use of previously computed grid points

In this and the following section we discuss acceleration techniques for the computation of the escape time. For simplicity the discussion is restricted to the real-valued escape-time function $E_R(x)$. It is straightforward to accelerate the weighed average escape time computation by the same methods.

The first idea for speeding up the escape-time computation is to explicitly make use of the recursive nature of the definition of escape time. The typical situation is that

$$E_R(x) = 1 + \max_{k=1,\dots,N} E_R(F_k^{-1}(x)). \quad (12)$$

The evaluation of this formula becomes very fast, when the values $E_R(F_k^{-1}(x))$ are already available. In the other case, these values should be computed, again recursively, and *stored* so that at a later time they may be reused for computation of escape time at other points x . To make this a feasible technique, a value of E_R should be computed for centers of pixels in an image only. This amounts to replacing $E_R(F_k^{-1}(x))$ in formula (12) by $E_R(c)$, where c denotes the center of the pixel, in which the point $F_k^{-1}(x)$ falls. Thus, a roundoff error will be introduced. A more accurate approach is to interpolate the values of the escape-time function between grid points while evaluating (12). It is of advantage to compute the image in a spiraling way working from the outer parts towards the center. In that way we can expect to make the most use of stored values of the escape time. Experimental tests indicate an approximately twofold speedup for the examples considered in this article.

The method works well only if the whole attractor is included in the image: If only a small portion of it is within the image bounds, the array of stored values at grid points or pixel centers is of little use. The bulk of the needed values of $E_R(x)$ will be for points x not covered by the image, and storing values of $E_R(x)$ outside the image would require (too) much memory.

4.5 Acceleration by the use of point-to-point coherence

The escape-time method is based on building a tree of inverse transformations starting at a given grid point. For an image displaying the escape time, the tree has to be constructed for all pixels. The basic idea of a *continuation method* is to update the tree computed for the previous point rather than always recomputing it from scratch. Consequently, associated with each node is the compound transformation that takes the starting point to this node. Since all the compound transformations are affine, thus continuous, we do not expect the tree to change much due to small changes in x (when we move from one pixel to the next). Hence, we traverse the tree and check only the leaves. As we move

from one pixel to another, some branches may have to be cut, and some others to be extended, but both operations are quite easy to perform. This is why we expect that the incremental method for updating the tree will be faster than constructing it always from the scratch.

The tree of preimages associates the following information with each node:

1. The inverse affine transformation accumulated so far, i.e.

$$G^{-1} = F_{i_k}^{-1} \cdots F_{i_1}^{-1}. \quad (13)$$

2. The depth of the node in the tree, i.e. the number k in equation (13).
3. An array of N pointers to the nodes below the present node. The node is a leaf node, if G^{-1} maps the starting point x outside of the disk D_R , while the image of x under the composite transformation at the parent node is still in D_R . Thus, $\|F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x)\| \geq R$ and $\|F_{i_{k-1}}^{-1} \cdots F_{i_1}^{-1}(x)\| < R$. In this case there is no array of pointers for child nodes.

The algorithm for updating the tree is given as a pseudo code.

When building the tree, the following computations are performed for each node and each of the N affine transformations:

1. Composition of $F_{i_{k+1}}^{-1}$ and G^{-1} amounting to 20 floating point operations (flops).
2. Application of the compound map to x (8 flops).
3. Verification whether the resulting point is outside of the disk of radius R (3 flops and 1 comparison).

The updating technique requires the storing of the entire tree in memory (about 8 numbers and N pointers per node). Additionally, compositions of transformations that are not necessary in the basic approach must be made here. Therefore, our experimental tests do not indicate a large overall time-savings when this tree-updating method is used. It remains to be seen whether this negative result is an inherent property of the continuation method or an effect of its software implementation.

4.6 The relation between distance and escape time

In this subsection we study the limit behavior of the escape-time function $E_R(x)$ for $R \rightarrow \infty$. The larger the radius R , the longer are the sequences of preimages of a point $y \notin \mathcal{A}$ that remain in the disk D_R . In the limit we have *diverging point sequences* $(y_k)_{k=1,2,\dots}$ with $y_k = F_{i_k}^{-1} \cdots F_{i_1}^{-1}(y)$ and $\lim_{k \rightarrow \infty} \|y_k\| = \infty$. We may ask whether there exists a formula that compensates for the exploding norm of y_k and leads to a finite limiting value which somehow characterizes the initial point y relative to the diverging sequence.

This question is related to the iteration of the function z^2+c which also yields diverging point sequences $z_1 = z_0^2+c, z_2 = z_1^2+c, \dots$. The compensation for the increase in magnitude of z_k is derived from the fact, that in the limit the logarithm of $|z_k|$ doubles in each step as $k \rightarrow \infty$. Thus, the limit of $2^{-k} \log |z_k|$ exists. Moreover, its value is the potential function at the point z_0 , see (3).

Theorem 6 shows that in the case of IFS's the compensation for an application of an inverse affine transformation is the multiplication by the linear part of the affine transformation. The limiting quantity described by the compensation procedure is Euclidean distance to a point in the attractor.

Algorithm **Algorithm Update-escape-time-tree**($\mathcal{F}, tree, x, R, depth_{max}$)
Purpose A given tree of preimages of a point near x is modified to preimages of x
Input \mathcal{F} an IFS $\{F_1, \dots, F_N\}$
 $tree$ the given escape-time tree
 x a point not in \mathcal{A}
 R radius of sufficiently large disk
 $depth_{max}$ maximal depth of the tree
Global symbols N the number of transformations
Local symbols Id the identity transformation
 k integer
 \tilde{F}^{-1} inverse compound transformation at current node
 $depth$ depth of current node in the tree
 $child[]$ array of N pointers to child nodes of current node
Output $tree$ the updated tree

begin
 Traverse the tree and perform the following operations at each node.
 if (node is a leaf) **then**
 if ($\|\tilde{F}^{-1}(x)\| < R$ and $depth < depth_{max}$) **then**
 for $k = 1$ to N **do**
 $child[k] = \text{Build-escape-time-tree}(F_k^{-1}\tilde{F}^{-1}, depth)$
 end for
 else if ($\|\tilde{F}^{-1}(x)\| \geq R$) **then**
 replace the current node by its parent node
 while ($\|\tilde{F}^{-1}(x)\| \geq R$) **do**
 remove the obsolete node and all branches below
 replace the current node by its parent node
 end while
 end if
 end if
end

Procedure **Build-escape-time-tree** ($\tilde{F}^{-1}, depth$)
Purpose Create a subtree of preimages starting at an initial node with
 initial inverse map \tilde{F}^{-1} and depth $depth$
Input \tilde{F}^{-1} initial composite of repelling maps
 $depth$ depth of initial node
Local symbols k integers
 $child[]$ array of N pointers to child nodes
Output pointer to the root node of the computed subtree

begin
 if ($depth \geq depth_{max}$ or $\|\tilde{F}^{-1}(x)\| \geq R$) **then**
 return (Pointer to $(\tilde{F}^{-1}, depth, \text{NULL})$)
 else
 for $k = 1, \dots, N$ **do**
 $child[k] = \text{Build-escape-time-tree}(F_k^{-1}\tilde{F}^{-1}, depth + 1)$
 end for
 return (Pointer to $(\tilde{F}^{-1}, depth, child[])$)
 end if
end

Theorem 6 Let $\mathcal{F} = \{F_1, \dots, F_N\}$ be an IFS with invertible contractions,

$$\begin{aligned} F_i : \mathbf{R}^n &\rightarrow \mathbf{R}^n \\ x &\mapsto A_i x + B_i \end{aligned}$$

where $i = 1, \dots, N$. Furthermore, let $F_{i_1}^{-1}, F_{i_2}^{-1}, \dots$ be a infinite sequence of inverse transformations from \mathcal{F} . Then there is a unique point y in the attractor \mathcal{A} of \mathcal{F} such that for all points $x \in \mathbf{R}^n$

$$\lim_{k \rightarrow \infty} (F_{i_1} \cdots F_{i_k})(x) = y \in \mathcal{A} \quad (14)$$

and

$$\lim_{k \rightarrow \infty} A_{i_1} \cdots A_{i_k} F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x) = x - y .$$

Proof. The first property is well known, see e.g. [10]. To prove the second part of the theorem we denote

$$\tilde{A}_k = A_{i_1} \cdots A_{i_k}, \quad \tilde{F}_k = F_{i_1} \cdots F_{i_k} .$$

The composite \tilde{F}_k of k affine transformations is itself an affine map with linear part \tilde{A}_k and translational part \tilde{B}_k , and therefore it can be expressed as

$$\tilde{F}_k(x) = \tilde{A}_k x + \tilde{B}_k .$$

We have $\tilde{B}_1 = B_{i_1}$ and the term \tilde{B}_k can be computed by induction for $k \geq 2$. The result is

$$\tilde{B}_k = F_{i_1} \cdots F_{i_{k-1}}(B_{i_k}) .$$

The inverse of \tilde{F}_k is

$$\tilde{F}_k^{-1}(x) = \tilde{A}_k^{-1}(x - \tilde{B}_k) .$$

Thus, for all $x \in \mathbf{R}^n$

$$A_{i_1} \cdots A_{i_k} F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x) = \tilde{A}_k \tilde{F}_k^{-1}(x) = x - \tilde{B}_k . \quad (15)$$

Following (14) we have

$$\lim_{k \rightarrow \infty} \tilde{B}_k = y$$

which concludes the proof. \square

Let us interpret this theorem starting from an arbitrary point $y \in \mathcal{A}$. There exists at least one sequence of affine maps F_{i_1}, F_{i_2}, \dots such that $(F_{i_k}^{-1} \cdots F_{i_1}^{-1})(y) \in \mathcal{A}$ for all $k = 1, 2, \dots$ (see [14]). When we apply the same sequence of inverse transformations to any point $x \in \mathbf{R}^n$ and carry out the compensation procedure (multiplying by the linear parts of the affine transformations), we obtain the vector $x - y$ as a result.

We now address the question, which sequence of preimages of a point $x \notin \mathcal{A}$ determines the escape time $E_R(x)$ as $R \rightarrow \infty$. Equivalently, we may ask, which point $y \in \mathcal{A}$ determines the escape time of x in the above sense. The following theorem shows, that for a certain class of IFS's it is always the point $y \in \mathcal{A}$ which is closest to x . In this case points with equal escape time (in the limit, as $R \rightarrow \infty$) also have equal distance to the attractor.

As a preparation for the theorem we need a Corollary.

Corollary 7 Assume the conditions of Theorem 6 and let again $F_{i_1}^{-1}, F_{i_2}^{-1}, \dots$ be a sequence of inverse transformations from \mathcal{F} . Then for any $x \in \mathbf{R}^n$ and any $k = 1, 2, \dots$ the following inequality holds:

$$d(x - A_{i_1} \cdots A_{i_k} F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x), \mathcal{A}) \leq c \cdot \rho_{\max}^k \quad (16)$$

The constant $c > 0$ is independent of x and ρ_{\max} is the maximal contraction ratio of the affine transformations F_1, \dots, F_N .

Proof. The claim follows from the proof of Theorem 6. Formula (15) yields

$$x - A_{i_1} \cdots A_{i_k} F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x) = F_{i_1} \cdots F_{i_{k-1}}(B_{i_k})$$

which is independent of x and tends to a point in the attractor (called y in Theorem 6) as $k \rightarrow \infty$. From this equation and the contraction of the transformations of \mathcal{F} we conclude that for all points $z_0 \in \{B_1, \dots, B_N\}$ and $z_1 \in \mathcal{A}$

$$\begin{aligned} d(x - A_{i_1} \cdots A_{i_k} F_{i_k}^{-1} \cdots F_{i_1}^{-1}(x), \mathcal{A}) &\leq d(F_{i_1} \cdots F_{i_{k-1}}(z_0), F_{i_1} \cdots F_{i_{k-1}}(z_1)) \\ &\leq \rho_{i_1} \cdots \rho_{i_{k-1}} d(z_0, z_1) \\ &\leq c \cdot \rho_{\max}^k \end{aligned}$$

where ρ_i denotes the contraction ratio $\rho_i = \rho(F_i)$, and the constant c can be chosen as

$$c = \frac{1}{\rho_{\max}} \sup\{d(z_0, z_1) : z_0 \in \{B_1, \dots, B_N\}, z_1 \in \mathcal{A}\}$$

which is finite due to the compactness of the attractor \mathcal{A} . □

Theorem 8 Let $\mathcal{F} = \{F_1, \dots, F_N\}$ be an IFS with invertible contractions

$$\begin{aligned} F_i : \mathbf{R}^n &\rightarrow \mathbf{R}^n \\ x &\mapsto A_i x + B_i \end{aligned}$$

Assume that the transformations in \mathcal{F} have the same contraction ratio $\rho < 1$ such that $\|A_i x\| = \rho \|x\|$, for all $i = 1, \dots, N$ and $x \in \mathbf{R}^n$. (Each A_i is a composition of an orthogonal matrix and a uniform scaling by ρ). The limit escape-time function, defined as

$$\begin{aligned} E_\infty : \mathbf{R}^n \setminus \mathcal{A} &\rightarrow \mathbf{R} \\ x &\mapsto \lim_{R \rightarrow \infty} \left(E_R(x) + \frac{\log R}{\log \rho} \right), \end{aligned}$$

is the logarithm of the distance function,

$$E_\infty(x) = \log_\rho d(x, \mathcal{A}) \quad (17)$$

for all $x \notin \mathcal{A}$.

Proof. Let $x \notin \mathcal{A}$ and $y \in \mathcal{A}$ be a point closest to x , i.e. $d(x, y) = d(x, \mathcal{A})$. Furthermore, let $F_{i_1}^{-1}, F_{i_2}^{-1}, \dots$ denote a sequence of inverse transformations such that

$$F_{i_k}^{-1} \cdots F_{i_1}^{-1}(y) \in \mathcal{A} \quad (18)$$

for all $k = 1, 2, \dots$. In order to compute $E_\infty(x)$ we consider without loss of generality (E_R is monotonic and continuous w.r.t. R) a sequence of radii R_m tending to infinity such that the corresponding escape times are integers, namely

$$E_{R_m}(x) = m .$$

Thus, by definition of the escape-time function, for each $m = 1, 2, \dots$ there is a sequence $F_{j_1^m}, \dots, F_{j_m^m}$ of m transformations such that

$$\|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\| = R_m$$

and for all other sequences F_{i_1}, \dots, F_{i_m} of m transformations we have

$$\|F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\| \geq R_m .$$

Specifically, for the sequence F_{i_1}, F_{i_2}, \dots satisfying formula (18), we obtain

$$1 \leq \frac{\|F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{\|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\|} \quad (19)$$

where $m = 1, 2, \dots$. Since $\|A_k(z)\| = \rho\|z\|$ for all $k = 1, \dots, N$ and all points z , we get

$$1 \leq \frac{\|F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{\|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\|} = \frac{\|A_{i_1} \cdots A_{i_m} F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{\|A_{j_1^m} \cdots A_{j_m^m} F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\|} .$$

From Corollary 7 and the triangle inequality we obtain for the denominator

$$\|A_{j_1^m} \cdots A_{j_m^m} F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\| \geq d(x, \mathcal{A}) - c \cdot \rho^m .$$

If m is sufficiently large, say $m > m_0$, then $d(x, \mathcal{A}) - c \cdot \rho^m > 0$ and, thus

$$1 \leq \frac{\|F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{\|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\|} \leq \frac{\|A_{i_1} \cdots A_{i_m} F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{d(x, \mathcal{A}) - c \cdot \rho^m} .$$

Form Theorem 6 it follows that the numerator of the right fraction above tends to $d(x, y)$ as $m \rightarrow \infty$. According to the assumptions at the beginning of the proof $d(x, y) = d(x, \mathcal{A})$. Since the denominator tends to the same number, we conclude that

$$\lim_{m \rightarrow \infty} \frac{\|F_{i_m}^{-1} \cdots F_{i_1}^{-1}(x)\|}{\|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\|} = 1$$

and, it follows that

$$\lim_{m \rightarrow \infty} \rho^m \|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\| = d(x, \mathcal{A}) . \quad (20)$$

By construction

$$\rho^m \|F_{j_m^m}^{-1} \cdots F_{j_1^m}^{-1}(x)\| = \rho^m R_m = \rho \left(E_{R_m}(x) + \frac{\log R_m}{\log \rho} \right) .$$

Passing with m to the limit and taking (20) under consideration, we conclude

$$\rho^{E_\infty(x)} = d(x, \mathcal{A}) . \quad \square$$

In summary we have shown that the computation of the escape time of x yields a sequence of maps $F_{i_1}^{-1}, F_{i_2}^{-1}, \dots$ which at the same time supplies a coding of the point $y \in \mathcal{A}$ which is closest to x . Moreover, formula (17) provides a direct relationship between escape time and distance.

An example satisfying the conditions of the theorem is the dragon curve, which is given by an IFS consisting of two transformations (see Appendix). Note that both transformations are compositions of scaling transformations (multiplication by $\rho = \frac{\sqrt{2}}{2} \approx 0.707$), rotations by $\frac{\pi}{4}$ or $\frac{3\pi}{4}$ and translation (only for the second transformation). Thus, the conditions of the corollary are satisfied, and the lines of constant limit escape-time values coincide with the lines equidistant from the fractal.

The line of arguments cannot be extended to the case where the transformations scale differently (with differing numbers ρ) in two dimensions.

We conclude this section with a conjecture regarding the escape-time function $E_R(x)$ for a general IFS as we let the radius R of the escape-disk grow to ∞ . Let us assume that among the transformations of the IFS there is a unique transformation, say F_1 , having the largest contraction ratio of all maps. In this case the point which is given by the fixed point of the least contractive transformation F_1 seems to completely determine $E_R(x)$ for all $x \notin \mathcal{A}$ and R large (in the sense of Theorem 6). More precisely, if $x \notin \mathcal{A}$, then the iteration of F_1^{-1} will generate the longest sequence of preimages of x remaining in a sufficiently large disk D_R . In the limit, points with equal escape time have equal distance to the fixed point of F_1 , which may be called the *least attractive* point of the attractor. For example, in the Barnsley fern this point is the tip of the fern, and a hint of the conjecture can be seen in the circular region about that point in Plate 11.

5 The potential function of IFS attractors

Our last method for visualization of an IFS attractor is based on notions from electro-dynamics. This approach corresponds closely to the potential functions used for Julia sets and the Mandelbrot set [4,12]. If μ denotes a (normalized) measure for the distribution of the electrostatic charge distributed over the attractor, then the electrostatic potential [3] at a point $y \notin \mathcal{A}$ is given by

$$\Psi(y) = \int_{\mathcal{A}} G_n(x, y) d\mu(x) \quad (21)$$

where

$$G_n(x, y) = \begin{cases} \|x - y\|^{2-n} & \text{if } n \geq 3 \\ \log \|x - y\|^{-1} & \text{if } n = 2 \end{cases}$$

and n is the dimension of the space \mathbf{R}^n in which the IFS operates. We consider two cases for the distribution μ :

1. uniform distribution over \mathcal{A} , which takes into account only the geometry of the attractor,
2. distribution corresponding to the invariant measure on \mathcal{A} which is given implicitly by choosing probabilities $p_1, \dots, p_N \geq 0$ with $p_1 + \dots + p_N = 1$ for the IFS $\{F_1, \dots, F_N\}$ [2,10].

In both cases feasible techniques for computing integrals over the attractor are given. The computation may use either a finite approximation of \mathcal{A} or an adaptive iterative scheme.

5.1 Integration using a fixed approximation

A fixed approximation at pixel level of \mathcal{A} may be obtained by the **adaptive-cut** method from Section 2. This data is sufficient for computation of the potential with respect to the uniform distribution. Given a list of m pixels $x_i, i = 1, \dots, m$ approximating \mathcal{A} , then the approximate value of the potential is given by a Riemann sum:

$$\Psi(y) \approx \frac{1}{m} \sum_{i=1}^m G_n(x_i, y) .$$

For the case of the distribution conform with the invariant measure on \mathcal{A} additional weights are introduced:

$$\Psi(y) \approx \frac{\sum_{i=1}^m w_i G_n(x_i, y)}{\sum_{i=1}^m w_i} . \quad (22)$$

These weights must approximate the invariant measure on \mathcal{A} . One way to generate them uses the ergodic theorem of Elton [6], which states that with probability one the empirical distribution

$$\mu_k = \frac{1}{k} \sum_{i=1}^k \delta_{z_k}$$

converges weakly to μ as $k \rightarrow \infty$. In this formula $z_k, k = 1, 2, \dots$ is a sequence of points obtained by the stochastic algorithm (the chaos game), and δ_z denotes the Dirac measure located at point z . This implies, that the integral in (21) with μ replaced by μ_k in fact converges to the potential $\Psi(y)$ as $k \rightarrow \infty$. For the practical computation given by (22) the weights w_i of the pixels approximating \mathcal{A} are just frequency counts relative to the point sequence z_k generated by the chaos game.

5.2 Integration using an adaptive scheme

Since convergence of the process in (22) may be very slow (a large number of points from the chaos game may be necessary) an alternative procedure for the evaluation of the integral $\Psi(y)$ is introduced. In what follows we present a general algorithm suitable for computation of the integral

$$I(f) = \int_{\mathcal{A}} f d\mu \quad (23)$$

for any continuous function f . The method thus has a wide scope of applications of which the electrostatic potential is only one example.

The goal of the method is an approximation $I_\varepsilon(f)$ of $I(f)$ such that the error is bounded by a given tolerance $\varepsilon > 0$. The method is straightforward and based on the result that the invariant measure is a fixed point of the Markov operator M

$$\mu = M(\mu) = \sum_{i=1}^N p_i \mu F_i^{-1} . \quad (24)$$

Thus, we may replace μ in (23) by the sum on the right side. As the support of μ is \mathcal{A} , and $F_i^{-1}(x) \notin \mathcal{A}$ if $x \notin F_i(\mathcal{A})$, i.e. the support of μF_i^{-1} is $F_i(\mathcal{A})$, we obtain

$$I(f) = \sum_{i=1}^N p_i \int_{F_i(\mathcal{A})} f(x) d\mu F_i^{-1}(x).$$

Replacing again μ by the sum in (24) yields

$$I(f) = \sum_{i=1}^N \sum_{j=1}^N p_i p_j \int_{F_i F_j(\mathcal{A})} f(x) d\mu F_j^{-1} F_i^{-1}(x),$$

and continuing we get

$$I(f) = I^n(f) = \sum_{i_1, \dots, i_n=1}^N p_{i_1} \cdots p_{i_n} \int_{F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x) d\mu F_{i_n}^{-1} \cdots F_{i_1}^{-1}(x). \quad (25)$$

To obtain an approximation of $I(f)$ with prescribed tolerance $\varepsilon > 0$ it is sufficient to assume that each of the integrals in (25) is replaced by a term with an error bounded by ε . Note, that $\mu F_{i_n}^{-1} \cdots F_{i_1}^{-1}$ is a normalized measure with support $F_{i_1} \cdots F_{i_n}(\mathcal{A})$, thus

$$\inf_{x \in F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x) \leq \int_{F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x) d\mu F_{i_n}^{-1} \cdots F_{i_1}^{-1}(x) \leq \sup_{x \in F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x).$$

Continuity of f implies uniform continuity of f in \mathcal{A} , hence, there exists $\delta > 0$ such that $\|x - y\| < \delta$ implies $\|f(x) - f(y)\| < \varepsilon$ for all $x, y \in \mathcal{A}$. As all transformations $F_i, i = 1, \dots, N$ are strict contractions there is a number $n > 0$ such that the diameter of the set $F_{i_1} \cdots F_{i_n}(\mathcal{A})$ satisfies the inequality

$$\text{diam } F_{i_1} \cdots F_{i_n}(\mathcal{A}) < \delta$$

for all choices of indices i_1, \dots, i_n . With that we conclude that for any point $x_0 \in \mathcal{A}$ we have

$$\left| \int_{F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x) d\mu F_{i_n}^{-1} \cdots F_{i_1}^{-1}(x) - f(F_{i_1} \cdots F_{i_n}(x_0)) \right| < \varepsilon$$

thus,

$$I_\varepsilon(f) = \sum_{i_1, \dots, i_n=1}^N p_{i_1} \cdots p_{i_n} f(F_{i_1} \cdots F_{i_n}(x_0))$$

yields the desired approximation.

The above considered assumption for the approximation of the integrals in formula (25) is sufficient but not necessary. Therefore some modifications can be introduced that may speed up the computation.

1. Since the contraction ratios of the transformations F_i are generally not equal, the diameters of the sets $F_{i_1} \cdots F_{i_n}(\mathcal{A})$ may vary greatly. Therefore, an adaptive procedure is appropriate in the recursive computation of $I^n(f)$: As soon as the diameter of $F_{i_1} \cdots F_{i_n}(\mathcal{A})$ is less than δ , an approximation for the integral

$$\int_{F_{i_1} \cdots F_{i_n}(\mathcal{A})} f(x) d\mu F_{i_n}^{-1} \cdots F_{i_1}^{-1}(x)$$

is generated by $f(F_{i_1} \cdots F_{i_n}(x_0))$.

2. The maximal diameter δ may be adjusted to reflect the local properties of f . In particular for the computation of the potential function $\Psi(y)$, where we have for example $f(x) = \|x - y\|^{-1}$, a small value of δ is appropriate for sets $F_{i_1} \cdots F_{i_n}(\mathcal{A})$ close to y while a large value of δ is sufficient for sets farther away.

The color plate 13 shows the potential function for the Sierpiński gasket using a color map.

Conclusions

This paper presents methods for approximating and rendering sets defined by iterated function systems. They are divided into two basic categories: those that approximate the attractor and those that create numerical data that characterize the space outside the attractor. Examples include the Euclidean distance from the attractor, and the escape-time and potential functions. Various graphical techniques for representing the resulting fields of values are discussed. A close correspondence between methods for visualizing sets specified by IFS's and Julia sets is observed.

A number of problems are open for further research. Specifically, the escape-time method applies only to IFS's consisting of invertible transformations. An extension of this method is possible if a control mechanism allows only selected transformations to be applied to particular points of the attractor. This technique was applied to create Plate 11 in this paper, but details require further investigation. Another problem is related to a formal analysis of the complexity of various rendering algorithms. At the present time, only experimental data that relate speed of various algorithms are available.

Acknowledgements

The work on this paper has started while Przemyslaw Prusinkiewicz was visiting professor at the Institut für Dynamische Systeme, Universität Bremen, on invitation by Heinz-Otto Peitgen. Fritz von Haeseler, Hartmut Jürgens and Ralph Lichtenberger participated in many stimulating discussions. Craig Kolb from Yale University made available to us his ray tracer *Rayshade*, used to render many figures. The reported research has been supported by an operating grant, equipment grants and a scholarship from the Natural Sciences and Engineering Research Council of Canada. Facilities of the Department of Computer Science, University of Regina, and of the Institut für Dynamische Systeme, Universität Bremen were also essential. All support is gratefully acknowledged.

Appendix

We list the IFS codes for the examples, which are discussed in the main text. Here we use complex notation or homogeneous coordinates [7].

1. The dragon curve

$$F_1(z) = \frac{\sqrt{2}}{2}z \cdot e^{i\frac{\pi}{4}}$$

$$F_2(z) = \frac{\sqrt{2}}{2}z \cdot e^{i\frac{3\pi}{4}} + i$$

2. The Barnsley fern

$$T_1 = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.16 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \quad T_2 = \begin{bmatrix} 0.20 & 0.23 & 0.00 \\ -0.26 & 0.22 & 0.00 \\ 0.00 & 1.60 & 1.00 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} -0.15 & 0.26 & 0.00 \\ 0.28 & 0.24 & 0.00 \\ 0.00 & 0.44 & 1.00 \end{bmatrix} \quad T_4 = \begin{bmatrix} 0.85 & -0.04 & 0.00 \\ 0.04 & 0.85 & 0.00 \\ 0.00 & 1.60 & 1.00 \end{bmatrix}$$

3. The Sierpiński gasket in \mathbb{R}^2

$$F_1(z) = 0.5z$$

$$F_2(z) = 0.5z + 0.5$$

$$F_3(z) = 0.5z + 0.25 + 0.433i$$

4. The "star"

$$T_1 = \begin{bmatrix} 0.5000 & 0.0000 & 0.0000 \\ 0.0000 & 0.5000 & 0.0000 \\ -0.5000 & 0.0000 & 1.0000 \end{bmatrix} \quad T_2 = \begin{bmatrix} 0.5000 & 0.0000 & 0.0000 \\ 0.0000 & 0.5000 & 0.0000 \\ 0.5000 & 0.0000 & 1.0000 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 0.5000 & 0.0000 & 0.0000 \\ 0.0000 & 0.2500 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \quad T_4 = \begin{bmatrix} 0.3536 & 0.3536 & 0.0000 \\ -0.1768 & 0.1768 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

$$T_5 = \begin{bmatrix} 0.3536 & -0.3536 & 0.0000 \\ 0.1768 & 0.1768 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

5. The Sierpiński gasket in \mathbb{R}^3

$$T_1 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.500 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \quad T_2 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.500 & 0.000 \\ 0.500 & 0.000 & 0.000 & 1.000 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.500 & 0.000 \\ 0.250 & 0.433 & 0.000 & 1.000 \end{bmatrix} \quad T_4 = \begin{bmatrix} 0.500 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.500 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.500 & 0.000 \\ 0.250 & 0.145 & 0.409 & 1.000 \end{bmatrix}$$

References

- [1] Barnsley, M. F. and Demko, S., *Iterated function systems and the global construction of fractals*, Proceedings of the Royal Society of London Ser. A 399 (1985) 243–275
- [2] Barnsley, M. F., *Fractals Everywhere*, (Academic Press, San Diego, 1988)
- [3] Doob, J. L., *Classical Potential Theory and Its Probabilistic Counterpart*, (Springer-Verlag, New York, 1984)

- [4] Douady, A. and Hubbard, J., *Iteration des polynomes quadratiques complexes*, CRAS Paris 294 (1982) 123–126
- [5] Dubuc, S. and Elqortobi, A., *Approximations of fractal sets*, Journal of Computational and Applied Mathematics 29 (1990) 79–89
- [6] Elton, J. H., *An ergodic theorem for iterated maps*, Ergod. Th. & Dynam. Sys. 7 (1987) 481–488
- [7] Foley, J. D. and van Dam, A., *Fundamentals of Interactive Computer Graphics*, (Addison-Wesley, Reading, 1982)
- [8] Glassner, A. S. (ed.), *An Introduction to Ray Tracing*, (Academic Press, London, 1989)
- [9] Golub, G. H. and v. Loan, C. F., *Matrix Computations*, Second Edition, (Johns Hopkins, Baltimore, 1989)
- [10] Hutchinson, J. E., *Fractals and self-similarity*, Indiana Univ. Mathematics J., 30,5 (1981) 713–747
- [11] Mandelbrot, B. B., *The Fractal Geometry of Nature*, (W. H. Freeman and Co., New York, 1982)
- [12] Peitgen, H.-O. and Richter, P. H., *The Beauty of Fractals*, (Springer-Verlag, Heidelberg, 1986)
- [13] Peitgen, H.-O. and Saupe, D. (eds), *The Science of Fractal Images*, (Springer-Verlag, New York, 1988)
- [14] Prusinkiewicz, P. and Sandness, G., *Koch curves as attractors and repellers*, IEEE Computer Graphics & Applications 8,6 (1988) 26–40
- [15] Prusinkiewicz, P. and Sandness, G., *Attractors and repellers of Koch curves*, Graphics Interface '88
- [16] Reuter, L. Hodges, *Rendering and Magnification of Fractals Using Iterated Function Systems*, Ph. D. thesis, School of Mathematics, Georgia Institute of Technology (1987)
- [17] Stoer, J., *Einführung in die Numerische Mathematik I*, (Springer-Verlag, Heidelberg, 1979)
- [18] Williams, R. F., *Compositions of contractions*, Bol. Soc. Brasil. Mat. 2 (1971) 55–59

List of Plates

Plate 1 A three-dimensional extension of the Sierpiński gasket approximated by 21844 spheres.

Plate 2 Distance from the Sierpiński gasket computed using a finite point approximation rendered as a height field.

Plate 3 Four IFS's, the dragon curve, the fern, the Sierpiński triangle, and the "star" are compared using the distance estimator. Each object is rendered three times, with different values for the tolerance (resolution of the algorithm).

Plate 4 The distance function for the fern rendered as a height field.

Plate 5 A rendering of the dragon using spheres to represent the distance function.

Plate 6 A rendering of the fern using spheres to represent the distance function.

Plate 7 Index maps for the Sierpiński triangle. The color coding completely describes for each point the sequence of its preimages that will eventually determine the escape time. The first four indices I_1 to I_4 are used for the color coding, see Section 4.1.

Plate 8 The dragon curve with escape time rendered using color coding.

Plate 9 The dragon curve with escape time rendered as a height field.

Plate 10 The escape time for the Sierpiński triangle rendered as a height field. The radius R used for the computation of the escape-time function $E_R(x)$ is chosen as small as possible. The image demonstrates that the escape-time function is continuous, but not differentiable in this case.

Plate 11 Rendering of the escape-time function of the fern as a height field.

Plate 12 Comparison of the distance versus the escape-time function for a Cantor set. Top left: lines of constant distance from the attractor. Top right: lines of constant escape-time. Bottom left: plane partition showing which part of the attractor determines the distance value. Bottom right: the index map for the escape-time function.

Plate 13 Rendering of the electrostatic potential of the Sierpiński gasket using a color map.