COMPUTER DISPLAY OF LINEAR FRACTAL SURFACES

JOHN C. HART

B.S., Aurora University, 1987
M.S., University of Illinois at Chicago, 1989

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
and Computer Science
in the Graduate College of the
University of Illinois at Chicago, 1991

Chicago, Illinois

To my dear wife, Patty,

Who feels our baby's kicks.

Her contractions are not Lipschitz.

Instead they're *Braxton-Hicks.*

ACKNOWLEDGEMENTS

First and most importantly, I would like to thank my wife, Patricia A. Hart, for her selfless support toward this end. She suffered my absence for two long summers and has endured many lonely nights for this document. Therefore I dedicate this dissertation to her, as compensation for the many hours we could have otherwise spent together. In the same spirit I would like to thank the families of Joseph D. & Janice M. Hart, and Roger D. & Joyce M. Olson, for additional support during these years.

The members of the dissertation committee were selected for their influence on the research behind this document.

Its chairman: Thomas A. DeFanti, has supported me for the past four years with both salary and equipment. Tom's role as advisor far exceeded the scope of computer graphics.

Daniel J. Sandin and Louis H. Kauffman introduced me to 3-D fractals. Dan has been a constant companion in the research of fractal visualization algorithms. Lou has always encouraged the use of visualization in mathematics, resorting to pen and ink whenever CRT and pixels failed.

Maxine D. Brown has supported this research by communicating its results to more people than I will ever know. The breadth of Maxine's knowledge of visualization helped define the proper scope of this research.

Robert V. Kenyon's courses on computer graphics taught me more than I wanted to know about computer graphics. Only recently did I appreciate that.

Since he was first invited here in 1988, Alan Norton has been a valuable influence on this research. I interned at the IBM T.J. Watson Research Center during the Summer of 1989, during which time, Alan imparted the unpublished subtleties of fractal rendering.

I would also like to thank the staff and students of the Electronic Visualization Laboratory, for useful conversations and support, specifically: Craig Barnes, Sumit Das, Gordon Lescinsky, Gary Lindhal, Randy Hudson and Irv Moy. I would also like to thank the $(\mathrm{Art})^n$

lab at the Illinois Institute of Technology for promoting this work through PHSColograms.

My internship at AT&T Pixel Machines during the Summer of 1990 happened at a particularly depressing episode of their history. I am grateful for Steve Bourne's support of my internship there during such a critical time. There I enjoyed very informative conversations with Kamran Manoocheri, Jennifer Inman, Pete Segal and John Spicer at AT&T Pixel Machines, and with Don Mitchell and guests: John Amanatides and Paul Heckbert, at AT&T Bell Labs. I would also like to thank Frank & Lethe Lescinsky for housing me that summer.

Many other researchers in fractal geometry have been quite helpful, for which I would like to acknowledge: Daryl Hepting, Arnaud Jacquin, Benoit Mandelbrot, Ken Musgrave, Przemyslaw Prusinkewicz, Laurie Reuter, Dietmar Saupe, Richard Voss and Edward Vrscay.

JCH

PREFACE

The dissertation title "Computer Display of Linear Fractal Surfaces" is based on the overused "Computer Display of Curved Surfaces" which is found in the titles of many early works in computer graphics, in particular, the Ph.D. dissertations of J. Blinn and E Catmull. It is a satirical response to the overwhelming tendency in computer graphics to assume local smoothness of surfaces. Fractal geometry provides us with many examples of surfaces where this assumption fails. This failure has provided the author with a large set of open problems in computer graphics, for which this dissertation addresses and solves.

This dissertation was written using the $\mathcal{AMS}$-LaTeX document processing system. Its format meets the requirements of the University of Illinois at Chicago Graduate College Thesis Manual. The 2-D illustrations were created with Silicon Graphics Inc. IRIS-4D workstations, some using their product: "showcase." The 3-D illustrations were created on an AT&T Pixel Machine 964dX.

TABLE OF CONTENTS

List of Figures

SUMMARY

Historically, computer graphics algorithms have been optimized for locally smooth surfaces. These methods fail when the surface is very rough or, in particular, fractal. This dissertation outlines the shortcomings of current computer graphics techniques for rendering fractal shapes. It then proceeds in developing and describing new efficient techniques for rendering a subset of fractal surfaces called linear fractals.

The dissertation is separated into three sections. The first of these, Analysis, provides a mathematical framework for linear fractals. It begins with a formal treatment of iterated function systems, followed with a parallel discourse on recurrent iterated function systems. The iterated function system, along with its recurrent generalization, provides a fundamental theoretical model for linear fractals.

This part's third chapter compares and contrasts different definitions of the adjective "fractal." It concludes with a new definition, that of "locally fractal," which is justified with several examples. A discussion of linear and affine maps, the definition of linear fractal, and a summary of methods for measuring the dimension of linear fractals concludes the Analysis part of this dissertation.

This dissertation's second part, Modeling, describes techniques for creating linear fractals. It begins with a comparison of implicit and explicit models, showing, algorithmically, how the recurrent iterated function system model can be treated as both. The next chapter discusses the collage theorem as a philosophy for modeling linear fractals. The third chapter details three interactive methods for using the collage theorem to model linear fractals, whereas the final chapter discusses current methods for automatically modeling objects as linear fractals, determining the recurrent iterated function system computationally.

The third part — the main part — of this dissertation, Rendering, describes methods for displaying the linear fractal described by a recurrent iterated function system. It begins with an outline of current methods, showing their shortcomings when applied to fractals in

general.

The next four chapters of this part are devoted to describing methods for determining occlusion — solving the hidden surface problem. Ray tracing is the technique employed for this end, and efficient methods for finding the intersection of a ray with a linear fractal are developed. This includes determining the necessary level of detail by deriving the size of a pixel in object space, and computing an initial set that optimally contains the linear fractal.

Next, a shading method is developed, designed to allow the viewer to perceive fractal surface orientation, though not intended to be analytically correct. Finally, a chapter on the antialiased rasterization of linear fractal silhouettes, often fractals themselves, concludes this part of the dissertation.

A conclusion exhibiting the results of these methods, and suggesting further research follows. Two appendices, one reviewing basic metric space theory, the other documenting the recurrent iterated function system codes used to generate the illustrations that decorate this dissertation, are found at the end of this dissertation.

LIST OF ABBREVIATIONS

$\forall$	For all.
$\emptyset$	The empty set $\{\}$.
$\cup, \sqcup, \cap$	Union, disjoint union, intersection.
$\subset, \supset$	Subset, (note: $A \subset A$) superset.
2-D, 3-D	Two-dimensional, three-dimensional.
$(a,b), [a,b]$	The open interval $\{x : a < x < b\}$, the closed interval $\{x : a \leq x \leq b\}$.
$\overline{A}, \mathring{A}, \partial A$	The closure, interior, and boundary of A.
$A \setminus B$	The points in A that are not also in B.
$A + \epsilon$	The thickening of A by the ball of radius ϵ.
$a, A, \mathbf{A}, \mathcal{A}, \mathbb{A}$	The point a, the set A, the N-tuple of sets $\mathbf{A}$, the collection of sets $\mathcal{A}$, the space $\mathbb{A}$.
$B_r(x)$	The closed ball of radius r centered at x.
$\mathcal{C}$	The set of nonempty compact subsets of space.
$\mathbb{C}$	The complex plane.
card$\cdot$	Number of elements in a set.

CFG	Context-free grammar.
CSG	Constructive solid geometry.
fBm	Fractional Brownian motion.
FFT	Fast-Fourier transform.
G_v, G_e	Vertices and edges of digraph G.
IFS	Iterated function system.
L-system	Deterministic Lindenmeyer parallel context free grammar.
M^T, M^{-1}	The transpose, and inverse, of matrix M.
MFLOPS	Million floating-point operations per second.
MIPS	Million instructions per second.
$\mathcal{P}(\cdot)$	The power set (collection of all subsets) of a set.
$\mathbb{R}, \mathbb{R}^n$	The real numbers, n-dimensional Euclidean space with the implied distance metric.
$R(\mathbf{o}, \vec{\mathbf{d}})$	The ray extending from $\mathbf{o}$ in the direction of $\vec{\mathbf{d}}$.
RIFS	Recurrent iterated function system.
$U_r(x)$	The open ball of radius r centered at x.

$v_i, (v_i, v_j)$	The ith vertex and the edge from ith to jth vertex of G.
$w(), \mathbf{w}$	A contraction, a family of contractions.
$w_1 \circ w_2$	The composition $w_1(w_2())$.
$w^{\circ i}$	The i-fold composition of w.
$\mathbf{x}, \vec{\mathbf{x}}, x_i$	Homogeneous point, homogeneous vector, ith coordinate of point/vector $\mathbf{x}$.

Part I

INTRODUCTION

Chapter 1

OVERVIEW

This dissertation covers many topics. Its main contribution is in the rendering of linear fractal sets, in Part IV. It also provides a detailed treatment of iterated function systems and their recurrent extensions, and fractal geometry, in Part II. Part III, included for continuity, describes the various methods for using the mathematical structures from Part II to model objects, so they may be rendered by the methods in Part IV.

The specific goals of this dissertation are outlined as follows.

Part II develops the linear fractal model from a firm basis in topology and geometry. Chapter 3 introduces iterated function systems using the simplest results of set theory and metric spaces. Chapter 4 examines the recurrent iterated function system model, exposing subtleties. It is a parallel to Chapter 3. Chapter 5 properly defines "fractal," so that it embodies properties one associates with fractal sets. Finally, Chapter 6 clarifies the definition of a linear fractal.

Part III describes various techniques for modeling linear fractals, begining with an overview in Chapter 7. Chapter 8 characterizes the collage theorem philosophy of object modeling Chapter 9 documents interactive systems for modeling linear fractals. Finally, Chapter 10 reports recent breakthroughs in the automatic modeling of linear fractals.

Part IV begins by explaining the unique problems associated with rendering fractals in Chapter 11, which are solved in the rest of the chapters. Chapters 12, 13, 14, 15 and 16 together derive an efficient ray-linear fractal intersection method for hidden surface determination Chapter 17 simulate the shading of linear fractal surfaces, thereby portraying surface orientation. Finally, Chapter 18 reduces aliasing from rasterization with an efficient area sampling method, thereby eliminating the "jaggies."

Chapter 2

FRACTALS IN COMPUTER GRAPHICS

Fractals and computer graphics have historically enjoyed a symbiotic relationship, linking two relatively distinct areas of computer graphics: visualization and image synthesis.

2.1 From Visualization to Image Synthesis

Initially, a new fractal model is formulated and computer graphics techniques are then used to understand it better. The "visualization" of fractal structures has greatly increased their popularity and study. B. Mandelbrot's first book [Mandelbrot, 1977] describes, by way of concise visual analogy (initiator/generator pairs), the essence of fractal geometry. This one volume caused a resurgence of interest in this forgotten area of mathematics, directly inspiring the authorship of [Hutchinson, 1981] and [Falconer, 1985], to name a few. One rarely finds a paper on fractal geometry that lacks detailed computer-generated figures.

As fractal models become better understood, through visualization, they become easier to control. Controllable fractal models are widely used for "image synthesis," simulating

textures and forms that vary from the natural phenomena one encounters everyday to the novel constructions of mathematics.

The symbiosis continues: when new fractal models are discovered, they find applications in computer graphics. As computer graphics becomes better equipped to render these models, these and other fractals become easier to visualize, study and control.

2.2 Categorizing Fractals

Fractal models may be separated into two families: random and deterministic, based on their construction. The generation of both kinds of fractals may depend on streams of random numbers; the distinction is based on the influence these random numbers have on the shape. Altering the stream of random numbers will change the shape of a random fractal but will not affect the shape of a deterministic fractal.

2.2.1 Random Fractals in Computer Graphics

Random fractals are used for the simulation of natural phenomena. The basic random fractal model is a generalization of Brownian motion called "fractional Brownian motion" (abbreviated: fBm) [Mandelbrot & Ness, 1968].

Early researchers observed a resemblence between the graphs of fractional Brownian motion (fBm) and mountain silhouettes [Mandelbrot, 1982b], suggesting that fBm would be a useful tool for the image synthesis of various kinds of terrain.

Soon, R. Voss was using fBm to create images of mountains, coastlines and clouds. Working in the frequency domain, he would construct a $1/f^{\beta}$ frequency distribution. Then, using a fast-Fourier transform (FFT), this noise was transformed from the frequency domain onto a 2-D grid of altitudes, called a height field. These 3-D points were then polygonized, assigned realistic colors denoting altitude, shaded and scan-converted (rasterized, removing

hidden surfaces) to create an image of the terrain [Voss, 1988].

Shortly thereafter, A. Fournier and D. Fussel, simultaneously with but independent of L. Carpenter, devised a linear time algorithm of generating fractal mountains that was faster than the FFT method [Fournier *et al.*, 1982]. Their new method recursively subdividing each polygon into four smaller polygons, displacing the altitude of new vertices. B. Mandelbrot severely criticized this method for its ignorance of the statistical properties of fBm and "creased" appearance [Mandelbrot, 1982a]. None the worse for wear, it still receives wide use due to its simplicity and speed.

J. Kajiya, and later C. Bouville, developed efficient ray-tracing algorithms for rendering such recursively subdivided fractal terrains by creating a bounding volume hierarchy. Kajiya used "cheesecake" extents (extruded triangles) to bound all possible subdivisions of a triangle [Kajiya, 1983]. Bouville achieved a tighter bound using ellipsoids [Bouville, 1985].

G. Gardner created clouds and foliage using fractally textured ellipsoids. There, random fractal textures not only affected color but also opacity. Like Voss, Gardner's textures were created by Fourier methods, though it seems Gardner was concerned more with the appearance, and less with the statistical properties, of the texture [Gardner, 1984].

G. Miller used a context-sensitive subdivision scheme to reduce the "creasing" effects of previous subdivision methods. He also devised an efficient ray-tracing method, capitalizing on the height-field nature of these terrain models [Miller, 1986].

K. Musgrave, with C. Kolb, created terrain models using more advanced basis functions, such as the noise functions specified in [Perlin, 1985]. Like [Miller, 1986], they also treated terrains as a height field, developing a fast rendering algorithm called "grid tracing" [Musgrave *et al.*, 1989].

2.2.2 Deterministic Fractals in Computer Graphics

Early illustrations of 3-D deterministic fractals were generated at coarse resolutions, conveying the basic rules of construction; pictures of the final limit set were left to the viewer's imagination.

One of the first treatments of the special problems of rendering fractals was by A. Norton in 1982. His method of generating deterministic fractals, called "boundary tracking," was an early implementation of volume visualization. Boundary tracking generates a fixed-resolution voxel approximation of a fractal set. It runs in object space, precisely $O(n^d)$ space, where d is the box-counting dimension of the object. A z-buffer was used for hidden surface removal. The gradient of the z-buffer was used to simulate tangent planes along the surface, whose normal vector was used to diffusely shade the set [Norton, 1982; Norton, 1989b].

J. Holbrook studied deterministic fractals using a brute force $O(n^3)$ computation. He also used an $O(N)$ point enumeration technique to create a point cloud approximation, though they were so sparse he characterized them as "starfields" [Holbrook, 1983; Holbrook, 1987].

The point enumeration technique was developed further by L. Kauffman and D. Sandin. Shading and other depth cues were added to give the point cloud the perception of a 3-D surface. Unfortunately, the point clouds had large gaps that were difficult to fill in with this method [Hart, 1989; Sandin *et al.*, 1990; Hart *et al.*, 1990].

Later, a ray-tracing method was developed to efficiently render deterministic fractals. This method ran in image space and little more than image time, so close-up renderings of fractal surfaces took only slightly longer than far-away renderings [Hart, 1989; Hart *et al.*, 1989].

2.2.3 Linear Fractals in Computer Graphics

Linear fractals are a subset of deterministic fractals. They have the property that their construction rules are linear functions.

In 1984, A. R. Smith used formal languages to create linear fractal models he called "graftals" [Smith, 1984]. Graftals are also known as L-systems. Later, in 1988, P. Prusinkiewicz and others used L-systems to model many sophisticated botanical structures [Prusinkiewicz *et al.*, 1988; Prusinkiewicz & Hanan, 1989; Prusinkiewicz & Lindenmayer, 1990].

In 1985, iterated function systems were introduced to the computer graphics community, by one of their inventors: S. Demko, where they were used in 2-D for texturing and shape synthesis [Demko *et al.*, 1985]. The other inventor: M. Barnsley followed this up with by showing the power of iterated function systems as an image synthesis tool [Barnsley *et al.*, 1988]. There he alluded to their use in image compression but these methods have only recently been revealed [Jaquin, 1991].

The iterated function system model is commonly used in 2-D, sometimes as art [Lescinski, 1991]. Extensions to 3-D are mentioned in [Foley *et al.*, 1990] but they suggest the use of L-systems instead. The following quote, from [Demko *et al.*, 1985], calls for the development of graphics algorithms for 3-D iterated function systems:

> [Two-dimensional linear fractals] represent the simplest application of [iterated function systems]. For real objects, they depict only a kind of silhouette, but this in no way represents the limits of the technique. The extension of our methods to three-dimensional objects is the most important next step.

It wasn't until very recently, in [Hart & DeFanti, 1991], that computer graphics algorithms were developed specifically for modeling and rendering 3-D iterated function systems. Such algorithms are the subject of this dissertation.

Part II

ANALYSIS

Chapter 3

ITERATED FUNCTION SYSTEMS

The term "iterated function system" (abbreviated: IFS) was coined in [Barnsley & Demko, 1985] to describe a general framework of dynamics. However, most of the results about the IFS model were presented in [Hutchinson, 1981].

None of the theorems or proofs in this chapter are based on original ideas. However, the wording, layout and techniques used to make these theorems and their proofs coherent within a single chapter is new. Two examples: First, the proof of Theorem 3.4 is based entirely on the second half of the proof of the Blaschke selection theorem found in [Falconer, 1985]. Compare this short proof to the five part, two page proof of Theorem 2.7.1 in [Barnsley, 1988]. Second, the definition of "overlapping construction" from [Barnsley, 1988] only applies to connected sets and requires a difficult code space argument. Here, its definition is extended and uses only the simple terminology of metric spaces.

The term "iterated function system" is defined far from the beginning of the chapter. This placement makes it easier to explain why this definition may differ from definitions by other authors. Immediately following it is the proof that every IFS specifies a unique set, called its "attractor." This theorem is titled the "Fundamental Theorem of Iterated Function Systems," for it is the attractors that make iterated function systems interesting.

3.1 Lipschitz Functions, Contractions and Similtudes

A map is a function from one space into another. Most maps in this discussion take a metric space into itself. The category of map properties defined in this section describe the relationship of the distances between two points before and after a map is applied.

3.1.1 The Lipschitz Property

A map is "Lipschitz" when the ratio of the distances of the images of two points to that of the original two points is bounded by a constant. In other words, the images of points under a Lipschitz map can only get so far from each other, depending on how close they were before.

Definition 3.1 *A function w satisfies the* Lipschitz condition *on a metric space $(\mathbb{X}, d)$ if and only if there exists a finite non-negative s such that*

$$d(w(x), w(y)) \leq s d(x, y) \tag{1}$$

for all $x, y \in \mathbb{X}$.

The minimum value s such that (1) is true is called the *Lipschitz constant* of w and is defined

$$\mathrm{Lip} w = \sup_{x,y \in \mathbb{X}} \frac{d(w(x), w(y))}{d(x, y)}. \tag{2}$$

3.1.2 Contractions

Contractions are Lipschitz functions, with Lipschitz constant less than one. They bring points closer.

Definition 3.2 *A map w from metric space $(\mathbb{X}, d)$ to itself is a* contraction *if and only if*

$$d(w(x), w(y)) < d(x, y) \tag{3}$$

for all $x, y \in \mathbb{X}$.

In this context the Lipschitz constant is commonly called the *contractivity factor* of the map.

3.1.3 The Contraction Mapping Principle

The following result regarding contractions, the Contraction Mapping Principle, is a useful tool for existence proofs. It is an old result, with respect to the rest of this chapter, that can be found in any number of texts ([Kaplansky, 1977] for example). We will use it later to prove the Fundamental Theorem of Iterated Function Systems.

Theorem 3.1 (Contraction Mapping Principle) *Let* $(\mathbb{X}, d)$ *be a complete metric space and let* $w : \mathbb{X} \to \mathbb{X}$ *be a contraction on* $\mathbb{X}$. *Then* w *has a unique fixed point in* $\mathbb{X}$.

Proof: To show that at least one fixed point exists, pick any point $x \in \mathbb{X}$ and iterate w on it, creating the orbit

$$x_0 = x, \tag{4}$$

$$x_1 = w(x), \tag{5}$$

$$x_2 = w \circ w(x), \ldots. \tag{6}$$

The distances between successive points in this orbit can be characterized from (1) as

$$d(x_{i+1}, x_i) \leq s d(x_i, x_{i-1}). \tag{7}$$

where $s = \mathrm{Lip} w$. This gives, by induction,

$$d(x_{i+1}, x_i) \leq s^i d(x_1, x_0). \tag{8}$$

Thus $\{x_i\}$ is a Cauchy sequence and converges in a complete metric space.

Let $x \in \mathbb{X}$ be the limit point of the sequence. The point x is fixed under w since

$$\begin{aligned} w(x) &= w(\lim_{i\to\infty} x_i) && (9)\\ &= \lim_{i\to\infty} x_{i+1} && (10)\\ &= x. && (11) \end{aligned}$$

Now, suppose there were at least two distinct fixed points $x, y \in \mathbb{X}$. Then

$$d(w(x), w(y)) = d(x, y). \tag{12}$$

By (3), the map w cannot be a contraction. Thus, there can only be one fixed point of w. □

The following corollary is obvious from the previous theorem. It is explicitly declared here since it implies an important consequence of the Fundamental Theorem of Iterated Function Systems.

Corollary 3.2 *Let $w, \mathbb{X}$ be as in Theorem 3.1. Then all points in $\mathbb{X}$ converge to the same fixed point under iteration of w.*

Proof: This was shown for a specific x in the first half of the previous proof. Since this x was arbitrarily chosen, all points in $\mathbb{X}$ converge to a fixed point of w for which the second half of the proof shows there to be only one. □

3.1.4 Similtudes

Finally, if equality holds for (1) then the map is called a similtude.

Definition 3.3 *A map w from a metric space $(\mathbb{X}, d)$ to itself is a* similtude *if and only if there exists a finite non-negative s such that*

$$d(w(x), w(y)) = sd(x, y) \tag{13}$$

for all $x, y \in \mathbb{X}$.

Similtudes not only change the size of a set uniformly, but rotate, reflect and translate as well. The image of a set under a similtude is called a *similarity.*

3.2 The Hausdorff Metric

The Hausdorff metric is a useful tool in analyzing the IFS model. It measures the distance between two sets. For its definition, we will need the distance from a point to a set.

3.2.1 The Distance From a Point To a Set

The following function can be found in many metric space texts such as [Kaplansky, 1977]. It defines the distance from a point to a set as the distance from the point to it closest neighbor in the set.

Definition 3.4 *The distance $d(x, A)$ between the point $x \in \mathbb{X}$ and set $A \subset \mathbb{X}$ in the metric space $(\mathbb{X}, d)$ is given by*

$$d(x, A) = \inf_{y \in A} d(x, y). \tag{14}$$

3.2.2 The Distance Between Two Sets: The Hausdorff Metric

The following distance function for pairs of sets, the Hausdorff metric, was found useful in [Hutchinson, 1981], to prove the Fundamental Theorem of Iterated Function Systems.

Definition 3.5 *The* Hausdorff *metric h measures the distance between sets $A, B \subset \mathbb{X}$ as*

$$h(A, B) = \sup\{d(x, B), d(y, A) | x \in A, y \in B\}. \tag{15}$$

farthest from any point in B is found. Then the point $y \in B$ farthest from any point in A is found. The larger of these two maximal distances determines the Hausdorff distance between A and B.

3.2.3 The Thickening of a Set

The Hausdorff distance metric can be difficult to comprehend at first. The following definition and theorem make the Hausdorff distance easier to visualize.

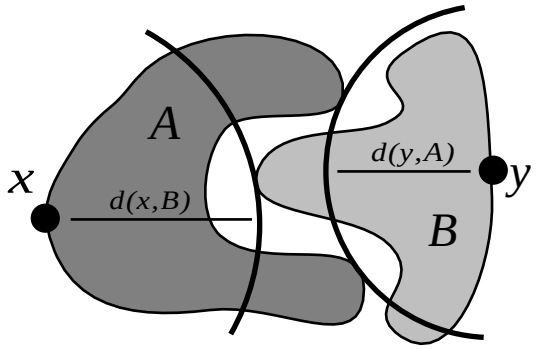

Figure 3.1: The Hausdorff metric between A and B.

We can "thicken" a set by including its surrounding points.

Definition 3.6 *The* thickening *of a set A by ϵ is the set*

$$A + \epsilon = \{x : d(x, A) \leq \epsilon\} \tag{16}$$

This is equivalent to a convolution of the set by a closed ball of radius ϵ (as it is sometimes defined).

The Hausdorff distance is given in terms of the smallest such thickening of one set that contains the other and vice-versa [Barnsley, 1988]. In some texts, such as [Falconer, 1985], this is how Hausdorff distance is defined.

Theorem 3.3 *For any two subsets A, B of metric space $(\mathbb{X}, d)$:*

$$h(A, B) = \inf\{\epsilon : A \subset B + \epsilon, B \subset A + \epsilon\}. \tag{17}$$

Proof: Let

$$\epsilon_B = \sup_{x \in A} d(x, B). \tag{18}$$

Then ϵ_B is the smallest value such that $A \subset B + \epsilon_B$ for if ϵ_B were any smaller, then some point in A would be farther from B than ϵ_B and so, would not be in its thickened version. Similarly, let

$$\epsilon_A = \sup_{y \in B} d(y, A). \tag{19}$$

Then ϵ_A is likewise the smallest value such that $B \subset A + \epsilon_A$. Looking at their maximum, we see

$$\begin{aligned} \epsilon &= \max\{\epsilon_A, \epsilon_B\} && (20) \\ &= \max\{\sup_{x \in A} d(x, B), \sup_{y \in B} d(y, A)\} && (21) \\ &= h(A, B). && (22) \end{aligned}$$

Hence the smallest value ϵ such that both $A \subset B + \epsilon$ and $B \subset A + \epsilon$ is precisely the Hausdorff distance of A and B. $\square$

In Figure 3.1, the Hausdorff metric was determined by $d(x, B)$. In Figure 3.2, the set B is thickened by $h(A, B)$. This thickening necessarily includes A by Theorem 3.3. Notice the difference in use of $d(x, B)$ between Figures 3.1 and 3.2.

3.2.4 The Complete Metric Space of Compact Subsets

One could construct a space whose "points" are sets from some other space. Then the Hausdorff distance could be used to make this space a metric space.

One possible space is $\mathcal{P}(\mathbb{R}^n)$, the set of all subsets of $\mathbb{R}^n$. However, the space $(\mathcal{P}(\mathbb{R}^n), h)$ is not a metric space, for $h(\mathring{A}, \bar{A}) = 0$ for all A. Since $\mathring{A} \neq \bar{A}$ for all but two sets in $\mathbb{R}^n$, h is not a valid metric on this space.

A subspace of $\mathcal{P}(\mathbb{R}^n)$ is $\mathcal{C}$, the collection of non-empty compact subsets of $\mathbb{R}^n$. Now $(\mathcal{C}, h)$ is a valid metric space since h meets the requirements for a metric on $\mathcal{C}$. Furthermore, we have the following classic result.

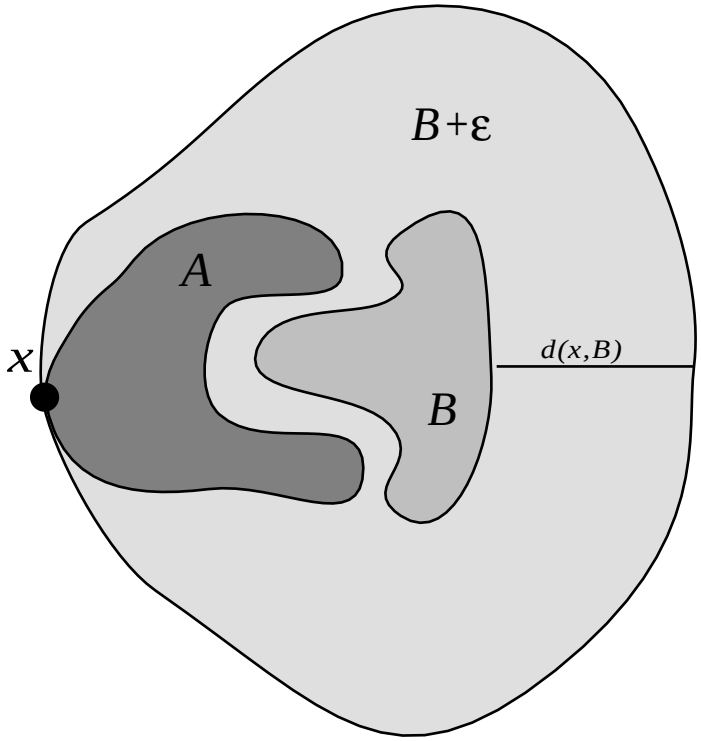

Figure 3.2: Set B, thickened by the minimal value $\epsilon = d(x, B) = h(A, B)$, so that $B + \epsilon$ contains A.

Theorem 3.4 *The metric space $(\mathcal{C}, h)$, where $\mathcal{C}$ is the collection of all non-empty compact sets of $\mathbb{R}^n$ and h is the Hausdorff metric, is complete.*

This short proof is based on the second half of the proof of the Blaschke selection theorem in [Falconer, 1985].

Proof: Let $\{K_i\}$ be a Cauchy sequence of compact sets $K_i \in \mathcal{C}$. That is, given an $\epsilon > 0$ there exists an N such that for any $i, j > N$,

$$h(K_i, K_j) < \epsilon. \tag{23}$$

Let

$$K = \bigcap_{j=1}^{\infty} \overline{\bigcup_{i=j}^{\infty} K_i} \tag{24}$$

which, like the K_i, is non-empty and, since it is the intersection of a decreasing sequence of compact sets, is compact. Given an $\epsilon > 0$ we have by (23) that there exists a j such that

$$\overline{\bigcup_{i=j}^{\infty} K_i} \subset K_j + \epsilon \tag{25}$$

and thus

$$K \subset K_j + \epsilon \tag{26}$$

and we are halfway there. Let $x \in K_j$, which, by (23), implies $x \in K_i + \epsilon$ for all $i \geq j$. Hence, if $k \geq j$ we have

$$x \in \left(\bigcup_{i=k}^{\infty} K_i \right) + \epsilon. \tag{27}$$

For each k there exists at least one point

$$y_k \in \overline{\bigcup_{i=k}^{\infty} K_i} \tag{28}$$

such that $|y_k - x| \leq \epsilon$. This forms a sequence $\{y_k\}$ of points from a compact space, with a subsequence necessarily converging to a point y, with $|y - x| \leq \epsilon$.

$$y \in \bigcap_{k=1}^{\infty} \overline{\bigcup_{i=k}^{\infty} K_i} = K \tag{29}$$

thus $x \in K + \epsilon$ and so

$$K_j \subset K + \epsilon, \tag{30}$$

which, with (26) and Lemma 3.3, means that $\{K_i\} \to K$ in the Hausdorff metric. □

3.3 The Hutchinson Operator

The Hutchinson operation applies a finite collection of maps to a set. The result is the union of the images of the original set under the maps in the collection.

3.3.1 Definition

Definition 3.7 *The* Hutchinson *operator of a finite set of maps* $\{w_i\}_{i=1}^N$ *on a set* A *is written* $\mathbf{w}(A)$ *and is defined*

$$\mathbf{w}(A) = \bigcup_{i=1\ldots N} w_i(A). \tag{31}$$

Suppose we have a collection of three similtudes of the form

$$w_i(x) = \frac{x + a_i}{2}, \qquad (i = 1, 2, 3) \tag{32}$$

which scales sets by one-half, then translates the set by $a_i/2$. The images of various sets under the Hutchinson operator of this collection are shown in Figure 3.3.

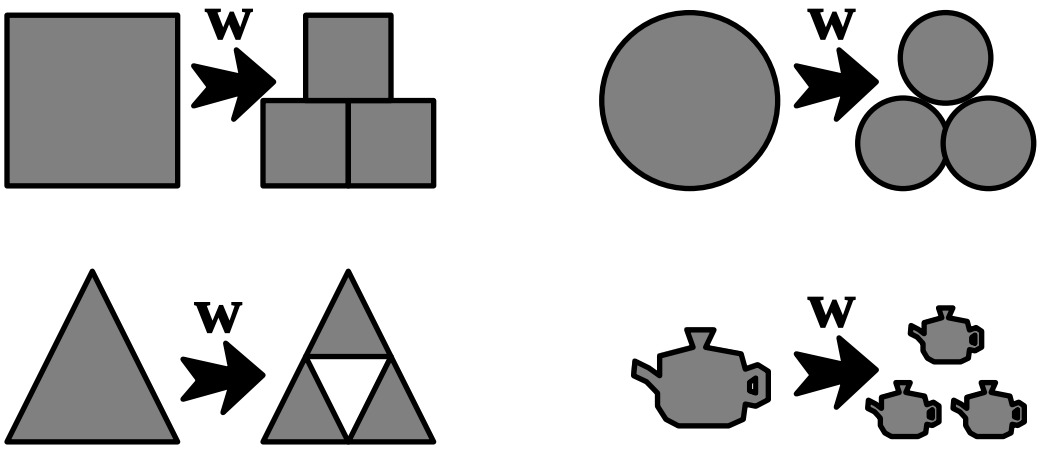

Figure 3.3: A Hutchinson operator applied to various initial sets.

3.3.2 Hutchinson's Lemma

The Hutchinson operator will be a convenient shorthand device for later proofs. It is also a contraction on the metric space of non-empty compact subsets.

Theorem 3.5 (Hutchinson's Lemma) *Let* $\mathbf{w}$ *be a finite collection of contractions* $w : \mathbb{R}^n \to \mathbb{R}^n$ *with Lipschitz constants* $\mathrm{Lip}w < 1$ *Then the Hutchinson operator* $\mathbf{w} : \mathcal{C} \to \mathcal{C}$ *is a contraction on the metric space* $(\mathcal{C}, h)$.

Proof: First, $\mathbf{w}$ is continuous; it takes compact sets to compact sets. Second, for any $A, B \in \mathcal{C}$,

$$\begin{aligned} h(\mathbf{w}(A), \mathbf{w}(B)) &= h\left(\bigcup_{i=1\ldots N} w_i(A), \bigcup_{j=1\ldots N} w_j(B) \right) && (33) \\ &\leq \sup_{i=1\ldots N} h(w_i(A), w_i(B)) && (34) \\ &\leq \mathbf{s}h(A, B) && (35) \end{aligned}$$

where

$$\mathbf{s} = \max_{i=1\ldots N} \mathrm{Lip}w_i \tag{36}$$

is the maximum Lipschitz constant of the contractions of $\mathbf{w}$. Since $\mathbf{s} \in [0, 1)$, $\mathbf{w}$ is a contraction on $(\mathcal{C}, h)$. □

This proof is also valid for countable collections of maps, so long as the Lipschitz constants of its maps are bounded well away from one. In other words, this requires a limit contractivity $\epsilon < 1$ where

$$\sup_{i=1\ldots\infty} \mathrm{Lip}w_i \leq \epsilon \tag{37}$$

3.4 Definition of Iterated Function System

Definitions of IFS vary. The most general descriptions list the metric space, a finite set of maps and a corresponding set of probabilities. For the sake of simplicity and clarity, the following abridged version will suffice for our purposes.

Definition 3.8 *An* iterated function system *is a finite collection* $\mathbf{w} = \{w_i\}_{i=1}^N$ *of contractions* $w_i : \mathbb{R}^n \to \mathbb{R}^n$.

This dissertation focuses on the geometry described by an IFS. To this end, the probabilities often associated with an IFS, which affect the distribution of points enumerated by the dynamical system an IFS describes, do not alter the resulting shape these points describe. Hence, the probabilities will not be used in the context of this dissertation.

Some authors do not require the maps of an IFS to be contractive. In the terms of these authors, the IFS defined in Definition 3.8 would be called "hyperbolic" [Barnsley & Demko, 1985]. We will be concentrating on linear maps (Chapter 6). If any map is not contractive in an IFS of linear maps, then the unique attractor of the IFS, as defined in the next section, will be the entire metric space — an undesirable result indeed.

3.5 The Fundamental Theorem of Iterated Function Systems

As mentioned in the previous section, an IFS describes a unique set: its attractor. The attractor is invariant under the Hutchinson operator of the IFS and is very often fractal. The following theorem, fundamental to the study of iterated function systems, asserts that, for any IFS, such a set always exists. It first appeared in [Hutchinson, 1981]. Its proof is relatively trivial given the earlier theorems of this chapter.

Theorem 3.6 (Fundamental Theorem of Iterated Function Systems) *For any IFS* $\mathbf{w} = \{w_i\}_{i=1}^{N}$ *there exists a unique non-empty compact set* $A \in \mathbb{R}^n$, *the invariant attractor of the IFS, such that*

$$A = \mathbf{w}(A). \tag{38}$$

Proof: The Hutchinson operator, $\mathbf{w}$, is a contraction (Theorem 3.5) on the complete metric space $(\mathcal{C}, h)$ (Theorem 3.4). Hence $\mathbf{w}$ has a unique invariant fixed point $A \in \mathcal{C}$ (Theorem 3.1). □

The unique fixed point of the Hutchinson operator of (32) is called "Sierpinski's gasket" [Mandelbrot, 1982b]. This attractor is shown in in Figure 3.4.

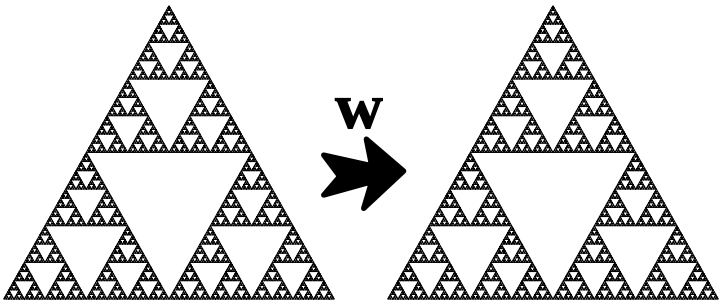

Figure 3.4: Sierpinski's gasket.

The 3-D version of this attractor is called "Sierpinski's tetrahedron" and is shown in Figure 3.5, rendered using the techniques from Part IV.

The following corollary explains the name "attractor," and can be found in [Hutchinson, 1981]. The unique form of its proof is due to the author.

Corollary 3.7 *Let* $\mathbf{w}$ *be an IFS with attractor* A*, and let* B *be any bounded non-empty set in* $\mathbb{R}^n$*. Then*

$$\lim_{i\to\infty} \mathbf{w}^{\circ i}(B) = A. \tag{39}$$

PROOF: Since B is bounded, there exists a compact set $B^+ \supset B$. Since B is non-empty, there exists a compact set $B^- \subset B$. By Corollary 3.2 and Theorem 3.6, we get the following loop implying equality,

$$A \subset \lim_{i\to\infty} \mathbf{w}^{\circ i}(B^-) \subset \lim_{i\to\infty} \mathbf{w}^{\circ i}(B) \subset \lim_{i\to\infty} \mathbf{w}^{\circ i}(B^+) \subset A. \square \tag{40}$$

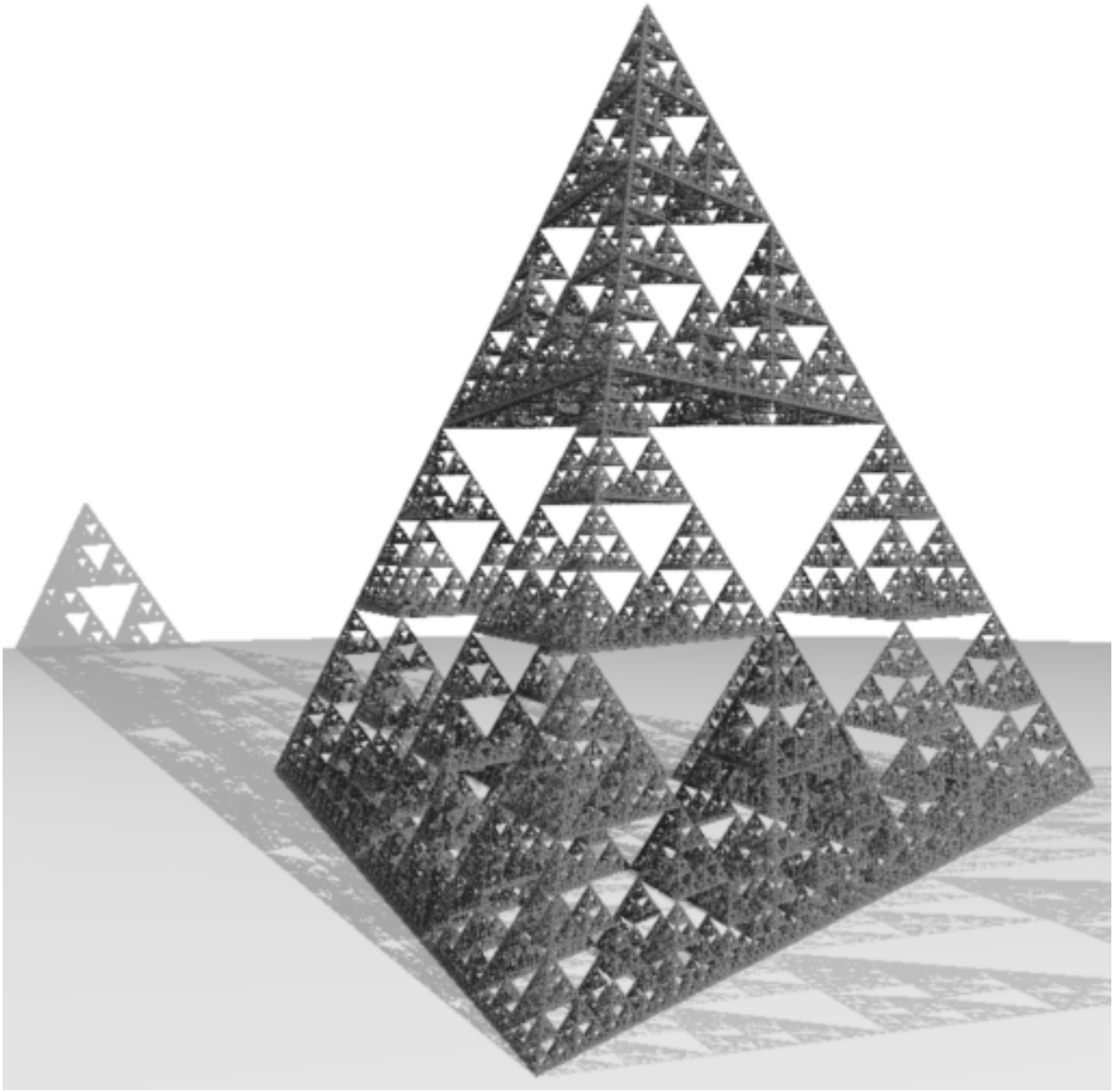

Figure 3.5: Sierpinski's tetrahedron.

3.6 The Open-Set Condition versus Overlapping Construction

One property of an IFS, defined in [Hutchinson, 1981], is the open-set property. It determines the uniqueness of the points in the attractor.

Definition 3.9 *An IFS* $\mathbf{w} = \{w_i\}_{i=1}^{N}$ *meets the* open-set *condition if and only if there exists an open set* $U \in \mathbb{R}^n$ *such that*

(a) $\mathbf{w}(U) \subset U$ *and*

(b) $w_i(U) \cap w_j(U) = \emptyset$ *when* $i \neq j$.

Iterated function systems with this property are often easier to analyze. Topological properties are simpler to compute on such attractors. Furthermore, many rendering methods are most efficient on attractors of iterated function systems with the open-set property.

An open-set IFS does not necessarily produce a disconnected attractor, such as the Cantor set in Figure 3.6 (left, black). In [Barnsley, 1988], IFS attractors that meet the open-set condition but are still connected are called "just-touching." Sierpinski's gasket is shown in Figure 3.6 to be open-set. It is an example of a just-touching attractor.

The overlapping construction of an attractor is the converse of the open set property. In [Barnsley, 1988], overlapping construction was defined only for connected attractors. This new definition extends the overlapping property to disconnected attractors, such as the overlapping Cantor set from [Falconer, 1987].

Definition 3.10 *The attractor* $A \in \mathbb{R}^n$ *of an IFS* $\mathbf{w} = \{w_i\}_{i=1}^{N}$ *is of* overlapping *construction if and only if* $\mathbf{w}$ *does not satisfy the open-set property.*

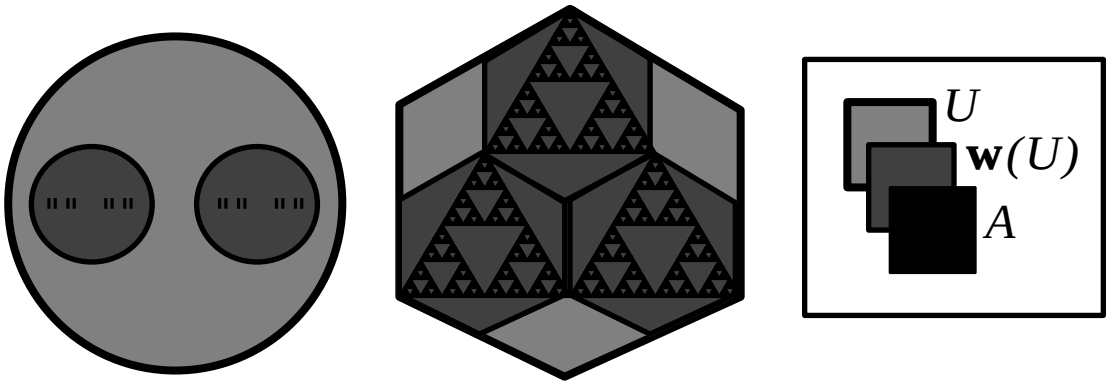

Figure 3.6: The Cantor set (left, black) and Sierpinski's gasket (middle, black) surrounded by open sets (light gray) which contain their images under the Hutchinson operator (dark gray). A legend appears on the right.

3.7 Disconnectedness

The following two theorems on disconnected attractors, and their proofs, are found in [Hutchinson, 1981]. They are good examples of pre-visualization — the ability to see certain topological properties of an attractor by inspection of its IFS.

3.7.1 Disjoint Images of the Attractor

Often one can determine if the attractor of an IFS is connected by a quick inspection of its contractions. If the contractions map the attractor into distinct components, then the attractor is disconnected.

Theorem 3.8 *Let* $\mathbf{w} = \{w_i\}_{i=1}^N$ *be an IFS with attractor* $A \subset \mathbb{R}^n$ *such that*

$$A = \bigsqcup_{i=1}^{N} w_i(A). \tag{41}$$

Then A is totally disconnected.

Proof: Let x, y be any two distinct points in A. By (41), x and y are either in the same component $x, y \in w_i(A)$ or in different components $x \in w_i(A), y \in w_j(A), i \neq j$. The first case shows x and y to be in disconnected components so we concentrate on repeated occurences of the second case under iteration. Let $d = |x - y|$. Furthermore, suppose x and y are both members of the same component $w_{i_k} \circ \cdots \circ w_{i_2} \circ w_{i_1}(A)$. But the diameter of this component is

$$\text{diam} w_{i_k} \circ \cdots \circ w_{i_2} \circ w_{i_1}(A) \leq s_{i_k} \cdots s_{i_2} s_{i_1} \tag{42}$$

which can be made arbitrarily small. Eventually this diameter will be less than $|x - y|$, excluding one of them from the connected component. Thus, x and y do not belong to the same connected component. Since x and y are arbitrary members of A, the set is totally disconnected. □

3.7.2 Sum of Lipschitz Constants

Furthermore, disconnected attractors arise when the contractivity factors of an IFS sum to less than one.

Theorem 3.9 *Let $\mathbf{w} = \{w_i\}_{i=1}^N$ be an IFS with attractor A such that*

$$\sum_{i=1}^{N} s_i < 1 \tag{43}$$

where s_i is the contractivity factor of w_i. Then A is totally disconnected.

Proof: From (43), we see that the sum of the diameters of the components is less than the diameter of the sum of components. Symbolically,

$$\text{diam}(A) > \text{diam}(A) \sum_{i=1}^{N} s_i = \sum_{i=1}^{N} \text{diam}(w_i(A)). \tag{44}$$

Let

$$\begin{aligned} \mathrm{diam}_n(A) &= \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_n=1}^{N} \mathrm{diam}(w_{i_1} \circ w_{i_2} \circ \cdots \circ w_{i_n}(A)) && (45) \\ &\leq \mathrm{diam}(A) \sum_{i_1=1}^{N} \sum_{i_2=1}^{N} \cdots \sum_{i_n=1}^{N} s_{i_1} s_{i_2} \ldots s_{i_n} && (46) \\ &= \mathrm{diam}(A) \left(\sum_{i=1}^{N} s_i \right)^n && (47) \end{aligned}$$

be the sum of the diameters of the components of the attractor. Equation (47) can get arbitrarily small given a sufficiently large n. Let x, y be two distinct points in A. Let n be large enough that $\mathrm{diam}_n(A) < |x - y|$. Then all of the components of the attractor, laid end to end, would still not be enough to connect point x to point y. Since this is true for all $x, y \in A$, A is totally disconnected. $\square$

Chapter 4

RECURRENT ITERATED FUNCTION SYSTEMS

The recurrent iterated function system (abbreviated: RIFS) model is an enhancement of the IFS model. Like the IFS model, a RIFS contains a finite set of contractions, but also restricts the composition of these maps. This enhancement greatly increases the power of this model, allowing more sets to be created using fewer maps.

Recurrent sets were first discussed in [Dekking, 1982] where an L-system model was described (see Chapter 7). The first thorough treatment of the RIFS model for their generation appears in [Barnsley *et al.*, 1989]. Several others have concurrently investigated this model under the adjective "recurrent" [Cabrelli *et al.*, 1991] and its aliases "Markov" [Womack, 1989], "controlled" [Prusinkiewicz & Lindenmayer, 1990], "language restricted" [Prusinkiewicz & Hammel, 1991] and "hierarchical" [Peitgen *et al.*, 1991]. The subtle distinctions between each of these models is described in this chapter shortly after we define RIFS. Nonetheless, these models describe the very same kind of restriction on map composition, which will be denoted using graph theory.

4.1 Graph Theory

The RIFS model requires a specification of allowable map compositions. A relation digraph[1] whose nodes correspond to maps and directed edges correspond to allowable compositions suffices and appears to be the most general description for restrictions of two map compositions. As such, we proceed with the definition of a digraph, differing slightly from [Deo, 1972].

4.1.1 Definition

Graphs are described here as an ordered pair of sets. The first is the set of vertices, the second, the set of edges. Edges are denoted as ordered pairs of vertices.

Definition 4.1 *A* digraph G *is a set of* N *vertices* $G_v = \{v_i\}_{i=1}^{N}$ *and a set of edges* G_e *which are ordered pairs* $(v_i, v_j), 1 \leq i, j \leq N$ *where* $(v_i, v_j) \in G_e$ *implies* G *contains a directed edge from* v_i *to* v_j.

Since G_e is a set, the same edge cannot appear twice in G. Thus the cardinality of G_e is at most the cardinality of G_v squared.

The number of edges into a vertex is called the "in-degree" of the vertex. Similarly, the number of edges out of a vertex is called the "out-degree" of the vertex.

4.1.2 Cycles

The following definition will help to define several properties of digraphs.

Definition 4.2 *A* directed (undirected) path *of vertices* $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ *connects vertex* v_{i_1} *to* v_{i_k} *if and only if for each pair of neighboring vertices* $v_{i_j}, v_{i_{j+1}}$ *the edge* $(v_{i_j}, v_{i_{j+1}}) \in G_e$ *(or the edge* $(v_{i_{j+1}}, v_{i_j}) \in G_e$*).*

[1] A digraph is a directed graph.

A cycle of edges is simply a path from a vertex to itself.

Definition 4.3 *Digraph G contains a* (directed, undirected) cycle *if and only if there exists a vertex $v_i \in G_v$ such that there exists a (directed, undirected) path of vertices in G_v connecting vertex v_i to itself.*

The ambiguous term cycle will imply directed cycle in this and following chapters. A cycle may be as simple as the edge (v_i, v_i). An "acyclic" digraph contains no cycles.

4.1.3 Strongly versus Weakly Connected Digraphs

We will need the following definitions for digraph connectedness, which agrees with [Deo, 1972].

Definition 4.4 *A digraph G is* (strongly, weakly)-connected *if and only if every pair of vertices $v_i, v_j \in G_v$ is connected by a (directed, undirected) path of vertices.*

We follow the convention, implied from this definition, that strongly-connected implies weakly-connected, which differs from [Deo, 1972]. Hence, a weakly-connected digraph may or may not be strongly-connected too.

If digraph G is weakly connected, then the cardinality of G_e is at least one less than the cardinality of G_v. If G is strongly connected, then the cardinality of G_e is at least the cardinality of G_v.

A strongly-connected digraph necessarily contains a cycle. A strictly weakly-connected digraph necessarily contains no cycle.

4.1.4 Condensation

Condensation "collapses" digraphs into simpler representative digraphs [Deo, 1972].

Definition 4.5 *Let $G = (G_v, G_e)$ be a digraph. Then the* condensation *of G is a digraph where each vertex corresponds to a unique maximal strongly-connected subgraph of G and each edge represents a set of edges from one maximal strongly-connected component of G to another.*

As noted in [Deo, 1972], the condensation of any strongly-connected digraph is a digraph containing a single vertex and no edges. Also, the condensation of a digraph contains no circuit.

4.2 Set N-tuples

The recurrent versions of the standard IFS analysis tools operate on N-tuples of sets. These will be denoted in bold as

$$\mathbf{A} = (A_1, A_2, \ldots, A_N). \tag{1}$$

A set N-tuple $\mathbf{A}$ of n-dimensional sets A_i belongs to the $n \times N$ dimensional space $(\mathbb{R}^n)^N$. Moreover, functions on these sets in this space are denoted with a superscripted N (i.e. $\mathbf{w} \to \mathbf{w}^N$).

One can think of $(\mathbb{R}^n)^N$ as N copies of $\mathbb{R}^n$ overlayed on top of each other. A set $\mathbf{A} \subset \mathbb{R}^2$ is better understood by drawing each of its N parts A_i on a separate clear sheet. Then the set $\mathbf{A}$ can be visualized in the space $(\mathbb{R}^2)^N$ by overlaying all N transparent sheets [Barnsley *et al.*, 1989].

The subset relation in $(\mathbb{R}^n)^N$ is defined as the logical "and" of the subset relation of its components.

Definition 4.6 *Let $\mathbf{A} = (A_1, A_2, \ldots, A_N)$ and $\mathbf{B} = (B_1, B_2, \ldots, B_N)$. Then $\mathbf{A} \subset \mathbf{B}$ if and only if $A_i \subset B_i$ for all $i = 1 \ldots N$.*

Notice that if $\mathbf{A} \subset \mathbf{B}$ then $\cup_i A_i \subset \cup_i B_i$.

4.3 The Recurrent Hausdorff Metric

The extension of the Hausdorff distance metric, as originally defined in [Barnsley *et al.*, 1989], combines the individual Hausdorff distances of the component sets using the chessboard metric.

Definition 4.7 *The* recurrent Hausdorff metric h^N *measures the distance between two subsets* $\mathbf{A} = (A_1, A_2, \ldots, A_N)$ *and* $\mathbf{B} = (B_1, B_2, \ldots, B_N)$ *of* $(\mathbb{R}^n)^N$ *as*

$$h^N(\mathbf{A}, \mathbf{B}) = \max_{i=1\ldots N} h(A_i, B_i). \tag{2}$$

As in the last chapter, we will construct a metric space of compact sets using the Hausdorff metric as a distance metric. Here, a set $\mathbf{A} \subset (\mathbb{R}^n)^N$ is compact if and only if every one of its parts A_i is compact in $\mathbb{R}^n$.

4.4 The Recurrent Hutchinson Operator

The recurrent Hutchinson operator, as first developed in [Barnsley *et al.*, 1989] is a generalization of the standard Hutchinson operator. It creates a new N-tuple as the selective union of images of sets from the original N-tuple.

Definition 4.8 *Let* $(\{w_i\}_{i=1}^N, G)$ *be an RIFS. Then the* recurrent Hutchinson operator $\mathbf{w}^N : (\mathbb{R}^n)^N \to (\mathbb{R}^n)^N$ *is defined*

$$\mathbf{w}^N(\mathbf{A}) = (\mathbf{w}_1(\mathbf{A}), \mathbf{w}_2(\mathbf{A}), \ldots, \mathbf{w}_N(\mathbf{A})) \tag{3}$$

where $\mathbf{A} = (A_1, A_2, \ldots, A_N) \in (\mathbb{R}^n)^N$ *and*

$$\mathbf{w}_j((A)) = \bigcup_{\{i:(v_i,v_j)\in G_e\}} w_j(A_i). \tag{4}$$

For example, consider the following four maps

$$w_i(x) = \frac{x + (\pm 1, \pm 1)}{2} \tag{5}$$

where $i = 1 \ldots 4$, taking all combinations of $(\pm 1, \pm 1)$. Each of these maps takes the closed square $[-1, 1] \times [-1, 1]$ to one of its quadrants.

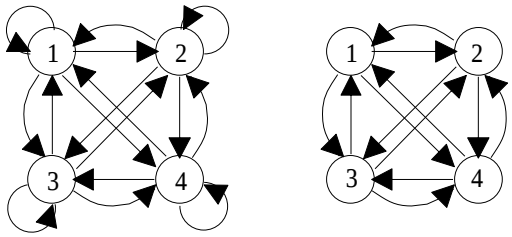

Figure 4.1: The complete digraph of four vertices and the corresponding digraph where the same vertex cannot be visited twice in a row.

Impose a restriction on the composition of these four maps, using the digraph in Figure 4.1 (right). In the latter case, the same map cannot be used twice in a row. The recurrent Hutchinson operation takes each one of four images of the square, reduces it by one-half, and places in the four quadrants of the original square *less* the image of the quadrant they are replacing. Figure 4.2 shows this generation sequence.

In Figure 4.2, the initial set is a square. One iteration of $\mathbf{w}^N$ produces the same square when the digraph in Figure 4.1 (left) is used, but produces only the shaded regions when the digraph in Figure 4.1 (right) is used. The empty boxes in Figure 4.2 denote the sections of the images under each of the maps that would appear under the standard Hutchinson operator, but do not appear under the recurrent Hutchinson operator.

As before, the recurrent Hutchinson operator is a contraction. However, this proof will need the following result. Its proof uses thickenings, but is otherwise based on the version

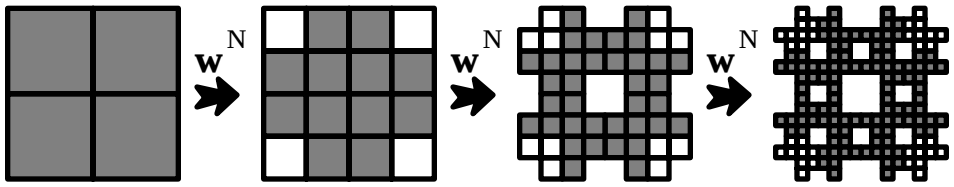

Figure 4.2: A recurrent Hutchinson operator applied to a square.

that appears in [Barnsley *et al.*, 1989].

Lemma 4.1 *For collections* $\mathcal{A} = \{A_i\}_{i=1}^N, \mathcal{B} = \{B_i\}_{i=1}^N$, *where* A_i, B_i *are subsets of metric space* $(\mathbb{X}, d)$,

$$h(\bigcup_{i=1\ldots N} A_i, \bigcup_{j=1\ldots N} B_j) \leq \sup_{i=1\ldots N} h(A_i, B_i). \tag{6}$$

Proof: Let

$$\epsilon = h(\bigcup_{i=1\ldots N} A_i, \bigcup_{j=1\ldots N} B_i). \tag{7}$$

Then

$$\bigcup_{i=1\ldots N} A_i \subset \bigcup_{j=1\ldots N} B_j + \epsilon, \tag{8}$$

For each $1 \leq i \leq N$ we have

$$A_i \subset \bigcup_{j=1\ldots N} B_j + \epsilon \tag{9}$$

and specifically

$$A_i \subset B_i + \epsilon_{B_i} \tag{10}$$

where ϵ_{B_i} is minimal. Still, $\epsilon_{B_i} \geq \epsilon$. Likewise, for each i there is a corresponding $\epsilon_{A_i} \geq \epsilon$. For each i the maximum thickening radius matches or exceeds the original thickening radius

$$\max\{\epsilon_{A_i}, \epsilon_{B_i}\} \geq \epsilon \tag{11}$$

as does its maximum over i. The proof is complete once the reader realizes that the left-hand side of (11) is the Hausdorff distance between A_i and B_i and the right-hand side is the Hausdorff distance between the union of A_i and the union of B_i. $\Box$

This argument is also valid for the maximum Hausdorff distance over countable collections of sets.

We may now state and prove a recurrent version of Hutchinson's Lemma.

Theorem 4.2 *Let $\mathbf{w}^N$ be the recurrent Hutchinson operator of RIFS $(\{w_i\}_{i=1}^N, G)$. Then $\mathbf{w}^N$ is a contraction on the metric space $(\mathcal{C}^N, h^N)$.*

Proof: Let $\mathbf{A} = (A_1, A_2, \ldots, A_N), \mathbf{B} = (B_1, B_2, \ldots, B_N) \in (\mathbb{R}^n)^N$. Consider the following chain of inequalities

$$\begin{aligned} h^N(\mathbf{w}^N(\mathbf{A}), \mathbf{w}^N(\mathbf{B})) &= \max_{j=1\ldots N} h(\mathbf{w}_j(\mathbf{A}), \mathbf{w}_j(\mathbf{B})) && (12)\\ &\leq \max_{i,j=1\ldots N} h(w_j(A_i), w_j(B_i)) && (13)\\ &\leq \max_{i,j=1\ldots N} s_j\ h(A_i, B_i) && (14)\\ &\leq \mathbf{s} \max_{i=1\ldots N} h(A_i, B_i) && (15)\\ &= \mathbf{s} h^N(\mathbf{A}, \mathbf{B}) && (16) \end{aligned}$$

where

$$\mathbf{s} = \max_{i=1\ldots N} \mathrm{Lip} w_i. \qquad (17)$$

Since $\mathbf{s} < 1, \mathbf{w}^N$ is a contraction on $(\mathcal{C}^N, h^N)$. $\Box$

4.5 Definition

As before, we have postponed definition the definition of RIFS so the reader will better understand its differences with other's definitions.

Definition 4.9 *A* recurrent iterated function system *is a finite set of contractions* $\{w_i\}_{i=1}^{N}$ *from* $\mathbb{R}^n$ *into itself, along with an* N*-vertex weakly-connected digraph* $G = (G_v, G_e)$ *containing some directed cycle from every vertex* $v_i \in G_v$ *back to itself.*

The digraph G is used to restrict map compositions. The iteration sequence $w_j \circ w_i$ is allowed if and only if a directed edge from vertex v_i to vertex v_j exists in G.

In order for the Chaos Game (Chapter 9) to work on an RIFS, it must be strongly connected. The rendering techniques developed in later chapters will work on any weakly-connected RIFS.

It is the connectedness of the digraph that differentiates the many different names for this IFS enhancement. In [Barnsley *et al.*, 1989], the term "recurrent," as applied to a row-stochastic matrix that described the digraph, implied the digraph was strongly connected, though they are careful to mention that $(\mathbf{w}, G)$ is a RIFS "whether or not [the row-stochastic matrix of G] is technically recurrent [i.e. G is strongly connected]." In [Womack, 1989], a RIFS was termed a "Markov iterated function system." A Markov IFS also used a strongly-connected digraph, also described by a row-stochastic matrix.

In [Reuter, 1987], a sunflower IFS attractor was distrubuted into a field of sunflowers using a second IFS. In [Peitgen *et al.*, 1991], this kind of RIFS is known as a "hierarchical iterated function system" which seems to imply a RIFS with a strictly weakly-connected digraph controlling map compositions. Hierarchical attractors have one IFS attractor at one level, then switch to another IFS at lower levels of detail.

In [Prusinkiewicz & Lindenmayer, 1990], a RIFS was called a "controlled iterated function system." Such an IFS was enhanced by a weakly-connected "control" graph. In [Prusinkiewicz & Hammel, 1991], a RIFS was described as a "language-restricted iterated function system." Here, the weakly-connected digraph G was described by a finite-state automaton where all nodes were "terminal."

4.6 The Fundamental Theorem of Recurrent Iterated Function Systems

As before, we consider it of fundamental importance that a unique set be associated with an RIFS. The following theorem and proof are based on [Barnsley *et al.*, 1989].

Theorem 4.3 *For any RIFS* $(\{w_i\}_{i=1}^N, G)$ *there exists a unique* N*-tuple* $\mathbf{A} \in (\mathbb{R}^n)^N$ *of non-empty compact sets such that*

$$\mathbf{A} = \mathbf{w}^N(\mathbf{A}). \tag{18}$$

Proof: As in the proof of Theorem 3.6, since $(\mathcal{C}^N, h^N)$ is complete (the finite product of complete spaces: Theorem 76 of [Kaplansky, 1977]) and $\mathbf{w}^N$ is a contraction on $(\mathcal{C}^N, h^N)$, the Contraction Mapping Principle implies that $\mathbf{w}^N$ posesses a unique fixed point in $(\mathcal{C}^N, h^N)$. □

The attractor of a RIFS is not an N-tuple, for we want to deal in an n-dimensional space, not an $n \times N$-dimensional space. Thus, we have, from [Barnsley *et al.*, 1989], the attractor of a RIFS.

Definition 4.10 *Let* $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N)$ *be a set such that it is invariant under the recurrent Hutchinson operator of a RIFS* $(\mathbf{w}, G)$,

$$\mathbf{w}^N(\mathbf{A}) = \mathbf{A}. \tag{19}$$

Then the attractor A *of the RIFS is given by*

$$A = \bigcup_{i=1\ldots N} \mathbf{A}_i. \tag{20}$$

The attractor of the IFS defined by the maps (5) and the digraph in Figure 4.1 (left) has the closed square $[-1, 1] \times [-1, 1]$ for an attractor. Imposing a restriction on the IFS of this

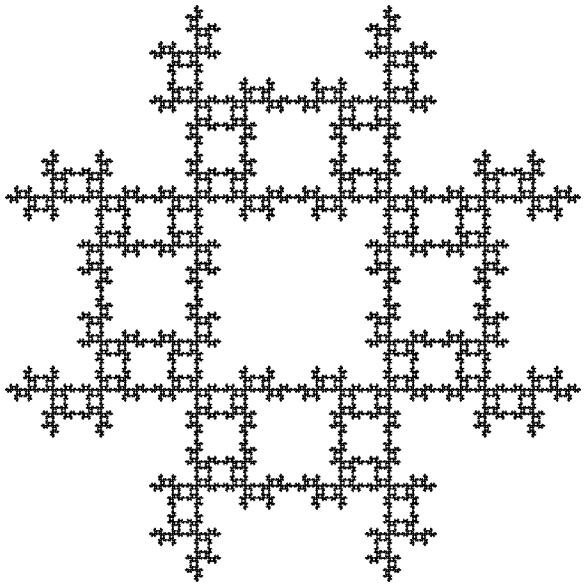

Figure 4.3: The fractal pound sign.

simple attractor can create a much more interesting set. Using Figure 4.1 (right) creates a RIFS whose attractor is shown in Figure 4.3.

The focus of this dissertation is on the rendering of such attractors in 3-D. The fractal pound sign can be extended into 3-D as an RIFS whose maps take the cube: $[-1,1] \times [-1,1] \times [-1,1]$ into its eight octants, and whose digraph does not allow the same map to be applied twice in a row. The resulting attractor in 3-D is shown in Figure 4.4.

Finally, we have the useful result that any initial set N-tuple will iterate to the attractor of an RIFS.

Corollary 4.4 *Let* $(\mathbf{w}, G)$ *and* $\mathbf{A}$ *be as in the previous theorem and let* $\mathbf{B}$ *be an* N*-tuple of non-empty bounded sets. Then*

$$\lim_{n\to\infty} (\mathbf{w}^N)^{\circ n}(\mathbf{B}) = \mathbf{A}. \tag{21}$$

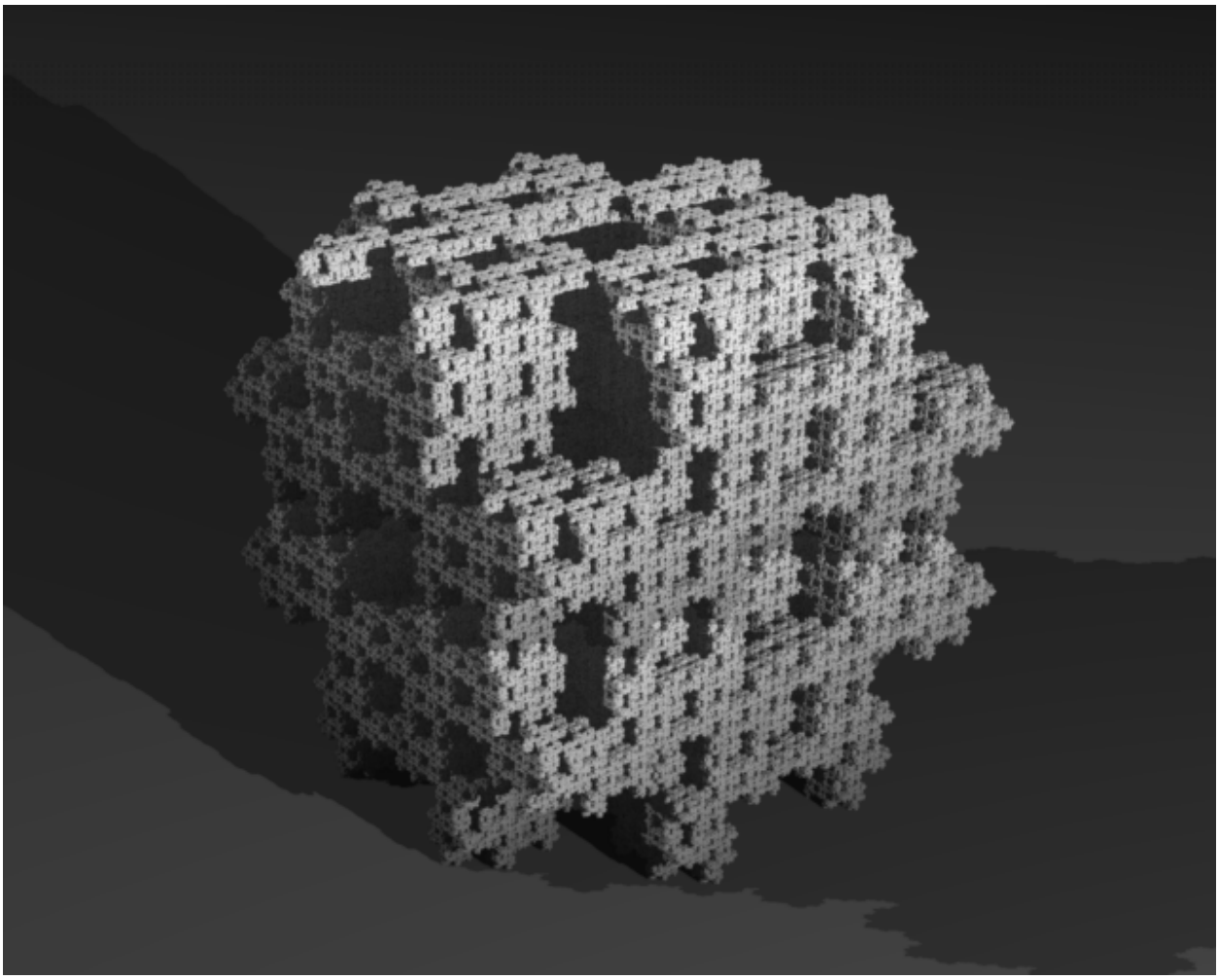

Figure 4.4: The 3-D fractal pound sign.

Proof: Since each component of $\mathbf{B}$ is bounded, there exists a compact set $B^+ \supset B_i$ for all $B_i \in \mathbf{B}$. Let

$$\mathbf{B}^+ = (B^+_, B^+, \ldots, B^+). \tag{22}$$

Since each component of $\mathbf{B}$ is non-empty, there exists a compact set $B_i^- \subset B_i$ for all $B_i \in \mathbf{B}$. Let

$$\mathbf{B}^- = (B_1^-, B_2^-, \ldots, B_N^-). \tag{23}$$

Then, by Corollary 3.2 and Theorem 4.3,

$$\mathbf{A} \subset \lim_{i \to \infty} (\mathbf{w}^N)^{\circ i}(\mathbf{B}^-) \subset \lim_{i \to \infty} (\mathbf{w}^N)^{\circ i}(\mathbf{B}) \subset \lim_{i \to \infty} (\mathbf{w}^N)^{\circ i}(\mathbf{B}^+) \subset \mathbf{A}. \tag{24}$$

As in the last chapter, they are all equal. □

Chapter 5

WHAT IT MEANS TO BE A FRACTAL

One of the most controversial areas of computer graphics in recent years is fractal geometry, characterized here as the study of forms of excessive dimension. Its progenitor, B. Mandelbrot, coined the word "fractal" publicly in 1975 to describe the recurring patterns across scale he had observed in natural phenomena.

5.1 Definitions of Fractal

One of the controversies of fractal geometry is the definition of fractal itself. The original definition compares a sets real Hausdorff dimension with it integral topological dimension [Mandelbrot, 1982b].

Definition 5.1 (The Original Definition) *A set A is fractal* if *its Hausdorff dimension is strictly greater than its topological dimension.*

Formally, this is an implication, not a definition, since the "only if" part is not included. It shows sufficiency, not necessity, for a set to be fractal. There are many sets that fall in the

so-called "gray area" of fractal geometry that fail to meet the requirements of this definition but are still recognized as fractal.

Currently, it is generally accepted that the "fractal" property needs no strict definition. The term "fractal" now simply refers to some kind of symmetry across scale.

One fundamental property of fractals is that they have detail at all magnifications. When this detail repeats, either exactly or statistically, the object posesses a symmetry across scale and is understood to be fractal. When this detail is asymetric across scale, then the object is probably better classified as a "multifractal" [Falconer, 1990].

Regardless of any controversies in the definition of fractal, it is still quite important to keep the Hausdorff dimension in mind when studying fractal sets, as shown most explicitly by the recent writings of K. Falconer, which inspired many of the chapters of this part of the dissertation.

Falconer's first book on the subject [Falconer, 1985], the so called "last book of Besicovich," the word "fractal" only appears in the preface, though the majority of the book is devoted to a thorough study of the topological properties of fractals. The main lesson of this book and its later rewriting [Falconer, 1990] is the importance of Hausdorff measure and dimension regardless of the current state of fractal geometry.

Except for the final section, this chapter is simply selections from [Falconer, 1985] and [Falconer, 1990]. The final section, Section 5.6, examines some counterexamples in fractal geometry: sets accepted as fractal but with equal Hausdorff and topological dimensions. The contradictions that arise from these counterexamples are settled with the proposal of a new definition of fractal — a new view of "what it means to be a fractal."

5.2 Hausdorff Dimension

The Hausdorff dimension is a generalization of the topological dimension from integers to real numbers. It is based on the Hausdorff measure, which is itself, a generalization of the

integral-dimensioned Lebesgue measure.

5.2.1 Definition

The first step is the definition of a δ-cover, which is a collection of sets that separate a set into δ- or smaller-sized sections.

Definition 5.2 *A collection $\mathcal{U}$ of sets $U \subset \mathbb{R}^n$ is a* δ-cover *of $A \subset \mathbb{R}^n$ if and only if*

$$A \subset \bigcup_{U \in \mathcal{U}} U, \tag{1}$$

($\mathcal{U}$ is a cover) and

$$0 < \operatorname{diam}(U) \leq \delta, \forall U \in \mathcal{U} \tag{2}$$

(the diameter of every element of $\mathcal{U}$ is smaller than δ).

The reader may recall the term "cover" from the definition of a compact set as one for which a finite open cover exists (Definition A.7). δ-covers alter this concept by allowing infinite covering sets, but of constrained size. Notice that any bounded set has a finite δ-cover for any $\delta > 0$. Thus δ-covers are used to show geometric, not topological, properties.

An "outer measure" is a positive real function on sets that is null on empty sets, smaller on subsets than on the whole. Furthermore, the outer measure of a union of sets is less than or equal to the sum of the outer measures of the individual sets. The Hausdorff measure is an outer measure.

Definition 5.3 *The* d-dimensional Hausdorff measure $\mathcal{H}^d(A)$ *of set A is*

$$\mathcal{H}^d(A) = \lim_{\delta \to 0} \inf \sum_{i=1}^{\infty} \operatorname{diam}(U_i)^d \tag{3}$$

where $\{U_i\}$ is a countable delta-cover of A.

The Hausdorff measure is similar to the Lebesgue measure (which is the basis of the integral) in that it is the sum of the sizes (to the dth power) of infinitesmal sets that, together, cover the set A. When $d = 1$, the Hausdorff measure is proportional to the length, $d = 2$ to area and $d = 3$ to volume. Hence, the Hausdorff measure is an extension of length, area and volume to real dimensions. In fact, some formulations of Hausdorff measure include a constant multiple so the n-dimensional Hausdorff measure equals the n-dimensional Lebesgue measure $\mathcal{L}^d$ (i.e. length, area, volume) for non-negative integers n.

Now, the length of a ball is infinite and the volume of a line segment is zero. Likewise, the d-dimensional Hausdorff measure of an e-dimensional solid will be infinite if $d < e$ and zero if $d > e$. When $d = e$ the Hausdorff measure may be zero, positive or infinite. This *critical* value $d = e$ of the Hausdorff measure on A is called the *Hausdorff(-Besicovich) dimension* of A.

Definition 5.4 *The* Hausdorff dimension *of set A is*

$$\dim A = \sup\{d : \mathcal{H}^d(A) = \infty\}. \tag{4}$$

Notice that also

$$\dim A = \inf\{d : \mathcal{H}^d(A) = 0\}. \tag{5}$$

5.2.2 Properties

The Hausdorff dimension posesses several properties that are useful in its estimation. The first is that the dimension remains invariant under scaling.

Theorem 5.1 *Let $A \subset \mathbb{R}^n$ and $s > 0$. Then*

$$\mathcal{H}^d(sA) = s^d\mathcal{H}^d(A) \tag{6}$$

where sA is the pointwise scaling of A by s.

Proof: Let $\mathcal{U}$ be a δ-cover of A. Then

$$\begin{aligned} \mathcal{H}^d_{s\delta}(sA) &\leq \sum_{U_i \subset \mathcal{U}} \operatorname{diam}(sU_i)^d && (7) \\ &= s^d \sum_{U_i \subset \mathcal{U}} \operatorname{diam}(U_i)^d && (8) \\ &\leq s^d \mathcal{H}^d_\delta(A) && (9) \end{aligned}$$

which, when taken to the limit as $\delta \to 0$, gives us

$$\mathcal{H}^d(sA) \leq s^d \mathcal{H}^d(A). \tag{10}$$

As so elegantly put in [Falconer, 1990], the other side is found simply by scaling sA by a factor of $1/s$. □

Another property is that the dimension of the union is the maximum of the dimensions. In fact, this is true for the union of countably many sets.

Theorem 5.2 *Let $\mathcal{A}$ be a countable collection of sets $A_i \subset \mathbb{R}^n$. Then*

$$\dim \bigcup_{A_i \in \mathcal{A}} A_i = \sup_{A_i \in \mathcal{A}} \dim A_i. \tag{11}$$

Proof: Let

$$A = \bigcup_{A_i \in \mathcal{A}} . \tag{12}$$

First, since $A_i \subset A$ for all i, then $\mathcal{H}^d(A_i) \leq \mathcal{H}^d(A)$ for all dimensions d. Thus $\dim A_i \leq \dim A$ for all i. Second, let $d > \dim A_i$ for all i. Then $\mathcal{H}^d(A_i) = 0$ for all i, so $\mathcal{H}^d(A) = 0$. Hence $\dim A \leq \sup_{A_i \in \mathcal{A}} A_i$. □

5.3 Box-Counting Dimension

When one sees the term "fractal dimension," more often than not, it is a reference to the box-counting dimension. The box-counting dimension is used in physics and other sciences because it can be easily computed on any kind of data.

5.3.1 Definition

The box-counting dimension is similar to the Hausdorff dimension in that it is found by covering a set with tiny "boxes" — actually any shape will do.

Definition 5.5 *The box-counting dimension* $\dim_B$ *of a set* A *is given by*

$$\dim_B A = \lim_{\delta \to 0} \frac{\log \text{card} \mathcal{U}}{\log \frac{1}{\delta}} \tag{13}$$

where $\mathcal{U}$ *is the* δ*-cover of* A *of minimal cardinality.*

One can compute the box-counting dimension of any set by using grids of various resolutions. Plotting the number of grid points the set intersects versus the resolution of the grid on a log-log grid will produce a line. If this line is straight then the box-counting dimension is the slope of this line [Mandelbrot, 1982b].

5.3.2 Relationship with Hausdorff Dimension

The box counting dimension of a set is always greater than or equal to the Hausdorff dimension of the set. They agree on well behaved sets.

Theorem 5.3 *Let* $A \subset \mathbb{R}^n$. *Then* $\dim_B A \geq \dim A$.

Proof: From the definitions of Hausdorff measure and box counting dimension we get the following inequality

$$\inf \sum_{i=1}^{\infty} \text{diam}^d(U_i) \leq |\mathcal{V}| \delta^d \tag{14}$$

for U_i is an element of δ-cover $\mathcal{U}$ and the infimum is over all such δ-covers, and $\mathcal{V}$ is a δ-cover of minimal cardinality. The δ's are the same, only the conditions of the covers have changed. The exponent d on both sides is the Hausdorff dimension of A.

Taking the limit as $\delta \to 0$ gives us the Hausdorff measure on the left hand side of (14). Theorem 5.1 insures that, by scaling, we can make the Hausdorff measure of a set arbitrarily

large without affecting its dimension. So, without loss of generality and for the sake of convenience, assume the Hausdorff measure of A is greater than one.

Taking the log of both sides then gives us

$$\log |\mathcal{V}| + d \log \delta > 0, \tag{15}$$

which means

$$\frac{\log |\mathcal{V}|}{-\log \delta} \geq d, \tag{16}$$

which can be easily rewritten into the desired result. □

One not-so-well-behaved set that demonstrates the problems of the box-counting dimension if shown in the following example (from [Falconer, 1990]).

Example 5.1 *The countable set* $A = \{1, \frac{1}{2}, \frac{1}{3}, \ldots, 0\}$ *has box-counting dimension* $\frac{1}{2}$.

Proof: Let $\mathcal{U}$ be a δ-cover of A. Then a different set from $\mathcal{U}$ is needed to cover each of the $i-1$ points: $1, \frac{1}{2}, \ldots, \frac{1}{i-1}$ of A, where

$$\delta = \frac{1}{i} - \frac{1}{i+1} = \frac{1}{i(i+1)}. \tag{17}$$

Thus, the minimum number of sets of diameter δ needed to cover A is i and we have

$$\frac{\log |\mathcal{U}_\delta|}{\log \frac{1}{\delta}} = \frac{\log i}{\log i(i+1)} \tag{18}$$

for minimal δ-cover $\mathcal{U}_\delta$. Taking the limit as $i \to \infty$ causes $\delta \to 0$ (17), and (18) converges to $\frac{1}{2}$. □

Clearly, countable sets have null Hausdorff dimension. Ordinarily, the box-counting dimension is otherwise quite useful.

5.4 Self-Similarity Dimension

The self-similarity dimension of a set is usually derived from its construction.

Definition 5.6 *If A is the attractor of an IFS satisfying the open set condition and with Lipschitz constants $s_i, i = 1 \ldots N$, then its self-similarity dimension* $\dim_S A$ *is given by the solution d of*

$$1 = \sum_{i=1}^{N} s_i^d. \tag{19}$$

Notice that if all similarities share the same Lipschitz constant s then the self-similarity dimension is simply

$$\dim_S = \frac{\log N}{\log \frac{1}{s}}. \tag{20}$$

Furthermore, if A is the attractor of an IFS of similtudes satisfying the open set condition, then the self-similarity dimension agrees with the Hausdorff dimension. Additionally, the Hausdorff measure of A is positive and bounded.

Theorem 5.4 *Let A be the attractor of an IFS of similtudes $\{w_i\}_{i=1}^N$ satisfying the open set condition. Then*

$$0 < \mathcal{H}^d(A) < \infty, \tag{21}$$

where $d = \dim_S A$. Specifically

$$\dim A = \dim_S A. \tag{22}$$

Proof: Let $s_i = \mathrm{Lip} w_i$ for each similtude of the IFS and let $d > 0$ be such that

$$\sum_{i=1}^{N} s_i^d = 1. \tag{23}$$

By repeated use of Theorem 3.6 we get

$$A = \bigcup_{1 \le i_1, i_2, \ldots, i_k \le N} w_{i_1} \circ w_{i_2} \circ \cdots \circ w_{i_k}(A). \tag{24}$$

Each of the k-fold compositions $w_{i_1} \circ w_{i_2} \circ \cdots \circ w_{i_k}$ has Lipschitz constant at most $s_{i_1} s_{i_2} \cdots s_{i_k}$. Thus

$$\sum_{i_k=1\ldots N} \cdots \sum_{i_1=1\ldots N} \mathrm{diam}(w_{i_1} \circ \cdots \circ w_{i_k}(A))^d \;\le\; \mathrm{diam}(A)^d \sum_{i_k=1\ldots N} \cdots \sum_{i_1=1\ldots N} (s_{i_1} \ldots s_{i_k})^d \tag{25}$$

$$= \text{diam}(A)^d \left(\sum_{i_1=1\ldots N} s_{i_1}^d \right) \cdots \left(\sum_{i_k=1\ldots N} s_{i_k}^d \right) \quad (26)$$

$$= \text{diam}(A)^d. \quad (27)$$

Let $\delta > 0$. Then k can be made sufficiently large so $\text{diam}(A) \leq \max_{i=1\ldots N} s_i^k \leq \delta$. Then $\mathcal{H}_\delta^d(A) \leq \text{diam}(A)^d$ which implies $\mathcal{H}^d(A) \leq \text{diam}(A)^d$ as $\delta \to 0$.

The other half of the proof, $\mathcal{H}^d(A) > 0$, is significantly more involved, and is beyond the scope of this chapter. It can be found in Section 8.3 of [Falconer, 1985]. □

5.5 Density Bounds and the Decomposition Theorem

This reiteration of [Falconer, 1985] concludes with a brief mention of density bounds.

Definition 5.7 *The density of a d-dimensional set $A \subset \mathbb{R}^n$ at a point $x \in A$ is given by the upper and lower bounds*

$$\overline{D}^d(A,x) = \overline{\lim}_{r\to 0} \frac{\mathcal{H}^d(A \cap B_r(x))}{(2r)^s}, \quad (28)$$

$$\underline{D}^d(A,x) = \underline{\lim}_{r\to 0} \frac{\mathcal{H}^d(A \cap B_r(x))}{(2r)^s}. \quad (29)$$

Set points where $\overline{D}^d = \underline{D}^d = 1$ are called *regular*; the other points are called *irregular.* A regular set is one that is regular at almost all (with respect to $\mathcal{H}^d$) points. Otherwise it is an irregular set. Manifolds are regular sets whereas fractals are irregular sets. However, there are some irregular sets that may not be considered fractal.

This is due to the definition of irregular as that which is not regular. This author believes that many of the problems with the definition of fractal can be solved by not defining fractal as that which is not Euclidean but by defining fractals more strictly and allowing sets to be both non-Euclidean and non-fractal.

The Decomposition Theorem, from [Falconer, 1985], states that a set's regular (Euclidean) parts and irregular (fractal) parts can be treated separately.

Theorem 5.5 *Let A be a subset of $\mathbb{R}^n$ such that $\dim A = d$. Then the set of regular points in A form a regular set and the set of irregular points in A form an irregular set.*

Proof: ... is beyond the scope of this chapter. See Chapter 2 of [Falconer, 1985]. □

5.6 Locally Fractal Sets

We conclude this chapter with some personal views justified with examples. Mandelbrot's original definition was a good one, but it fails on certain sets, and requires adjustment. Fractal are "sets with detail at every level of magnification," though not all such sets are fractal; as mentioned at the beginning of this chapter, some are better classified as multi-fractal, others are completely unstructured. We will attempt to form this statement into a new definition of fractal.

5.6.1 The Carrot Leaf

Let us start with the carrot leaf. It is a tree-like attractor: a stem with smaller carrot leaves extending from it. It is shown here in Figure 5.1.

Consider the straight line segments comprising its stem and, in fact, every visible branch of the carrot leaf. Magnifying any of these branches will not produce any more detail beyond some finite factor. Intuitively, this object does not seem fractal. In this case, intuition is supported by a mathematical argument.

Example 5.2 *Let A be the carrot leaf attractor. Then $\dim A = 1$. In particular, the carrot leaf is not fractal under the old definition.*

Proof: Let B_0 be the trunk of the carrot leaf and let B_1 and B_2 be the main branches, and so on. Let B be the countable union of all such branches B_i. Since the branches are line

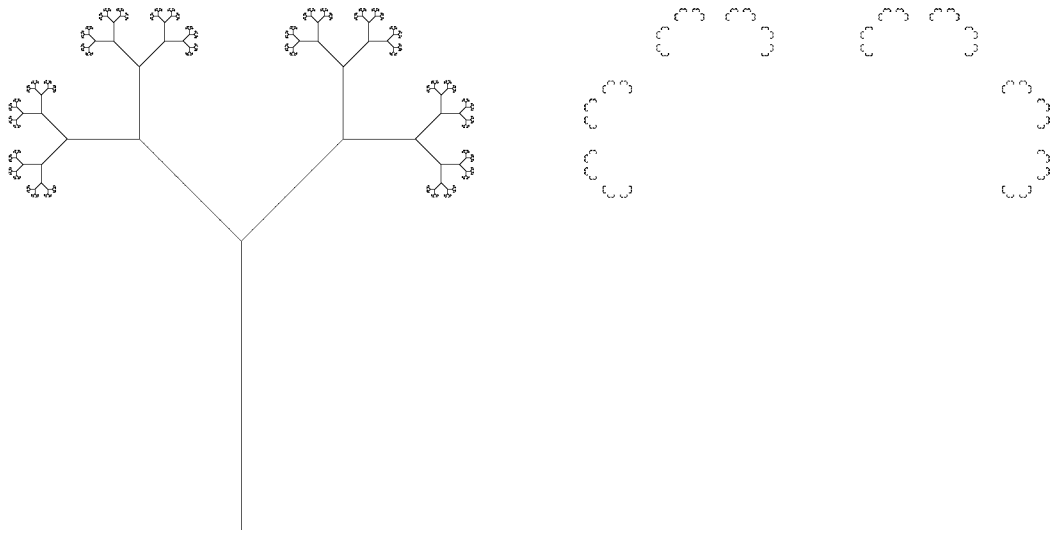

Figure 5.1: The "carrot leaf" attractor (left) and the set of its branching limit points (right).

segments,

$$\dim B = \sup_i \dim B_i = 1. \tag{30}$$

Let C be the limit points of branching structure B as illustrated in Figure 12.1. The set C is totally disconnected (visually confirmed by Theorem 3.8 and Figure 12.1) implying

$$\dim C < 1. \tag{31}$$

Since the carrot leaf consists of the (disjoint) union of its branching structure and its limit points

$$A = B \sqcup C \tag{32}$$

we have

$$\dim A = \max\{\dim B, \dim C\} = 1 \tag{33}$$

and the carrot leaf is not necessarily fractal under Definition 5.1. □

What we have done in this proof is separate the fractal part of the carrot leaf from the Euclidean part. One could say the carrot leaf is fractal only at its branch tips. In [Mandelbrot, 1982b], the branches were called a residue and such shapes whose fractal part's dimension is inferior to its topological part's were termed "subfractals."

This argument does not redeem Definition 5.1, for if the branches were lengthened so they intersected other branches, the resulting "fractal part" would be connected, with Hausdorff dimension greater than one, subsuming the Euclidean part and passing the original definition of fractal.

5.6.2 Locally Fractal

Clearly Definition 5.1 is the problem. The following definition of a fractal property called "locally fractal" is a step toward the proper definition of the property of fractal.

Definition 5.8 *A set $A \subset \mathbb{R}^n$ is locally fractal at point x if and only if*

$$\overline{D}^d(A,x) \neq \underline{D}^d(A,x). \tag{34}$$

Set A is locally fractal if and only if it is locally fractal at all of its points.

This would be the definition of irregular except for one important point. A set is irregular if at least one point is irregular whereas a set is locally fractal only if all of its points are irregular. "Locally fractal" is more exclusive than "irregular."

For the following chapters, fractals are hereby defined as sets that are locally fractal. This reflects its author's view of "what it means to be a fractal." It does not necessarily agree with definitions by other authors.

5.6.3 The Devil's Staircase

Define an accumulation function $A()$ as

$$A(x, B) = \frac{\text{card}([0, x] \cap B)}{\text{card}\, B}, \tag{35}$$

where B is any set in $\mathbb{R}$. Here cardinality is used in the loose sense, as the quotient of infinities is not well defined. In this case we are more concerned with the percentage of points in B less than or equal to x.

One could graph the accumulation function A versus x for the Cantor set C. Let $D \subset \mathbb{R}^2$ be the graph of $A(x, C)$. The set D is called the Devil's staircase [Mandelbrot, 1982b] and is shown in Figure 5.2.

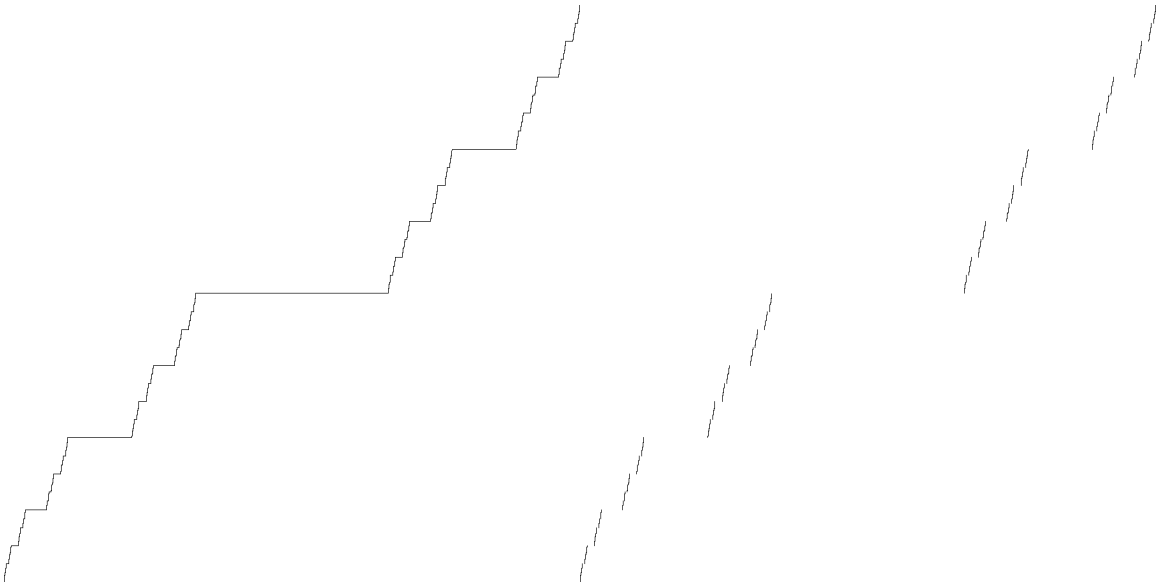

Figure 5.2: The Devil's staircase (left) and the set of its increments (right).

As reported in [Mandelbrot, 1982b], the Devil's staircase is rectifiable. In fact, its of length two. It is level at each of the countably many plateaus, corresponding to the gaps in

the Cantor set. The points where the graph changes levels are called its increments, which correspond one-to-one with the uncountable points of the Cantor set.

As before, this set is not necessarily fractal by Definition 5.1. The argument is essentially the same as the proof of Example 5.2.

Example 5.3 *Let $A \subset \mathbb{R}^2$ be the devil's staircase. Then* $\dim A = 1$

Proof: The horizontal intervals H_i in A correspond to the gaps of the Cantor set. Let

$$H = \bigcup_i H_i. \tag{36}$$

In A there are countably many such H_i so

$$\dim H = \sup_i \dim H_i = 1 \tag{37}$$

The remaining parts of A are a "skewed" Cantor set C whose dimension remains less than one (as shown in Figure 5.2). Hence we have

$$A = H \sqcup C, \tag{38}$$

giving

$$\dim A = \max\{\dim H, \dim C\} = 1 \tag{39}$$

and the devils staircase is not necessarily fractal under Definition 5.1. □

5.6.4 The Twindragon

The twindragon is a fractal shape that purportedly arose from a study of recursion [Mandelbrot, 1982b]. It is the "just-touching" union of smaller similarities, as shown in Figure 5.3. The twindragon is interesting because it can tile the plane, with a fractal border separating the tiles.

The twindragon is also not necessarily fractal under Definition 5.1. There is a subtle difference in this argument.

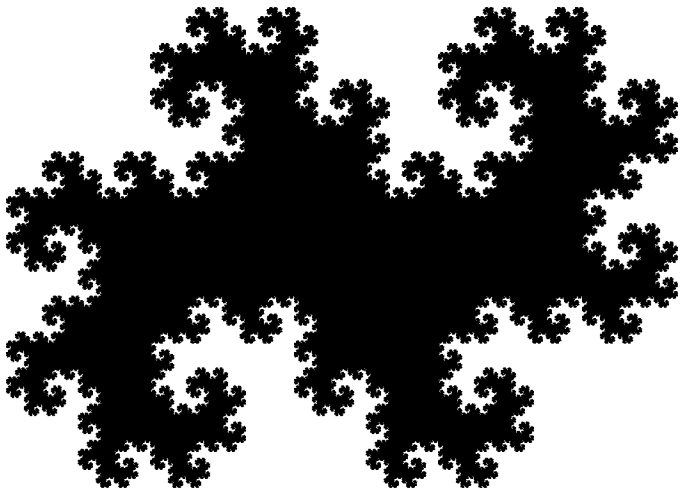

Figure 5.3: The twindragon.

Example 5.4 *Let $A \subset \mathbb{R}^2$ be the twindragon. Then* $\dim A = 2$.

Proof: The interior $\overset{\circ}{A}$ of the twindragon is open and non-empty. Its border ∂A is closed and in A. Thus,

$$A = \overset{\circ}{A} \sqcup \partial A. \tag{40}$$

The interior $\overset{\circ}{A}$ is open and bounded so contains and is contained in a ball with positive finite 2-dimensional Hausdorff measure. Thus, $\dim \overset{\circ}{A} = 2$.

The dimension of ∂A cannot exceed the dimension of the plane it is embedded in, so $\dim \partial A \leq 2$. Hence,

$$\dim A = \max\{\dim \overset{\circ}{A}, \dim \partial A\} = 2 \tag{41}$$

and the twindragon is not necessarily fractal under Definition 5.1. □

In fact, any set with non-empty interior is likewise not necessarily fractal under Definition 5.1. However, if it is locally fractal at its boundary points, then separating its boundary from its interior will produce a properly fractal set.

Chapter 6

LINEAR FRACTALS

Linear fractals are the product of RIFS's of affine maps. Such maps are used extensively in the area of computer graphics. This chapter begins by introducing linear and affine maps, and homogeneous coordinates, most of which can be found in [Foley *et al.*, 1990][1]. The next sections define linear fractals and show how they form naturally from cyclic object instancing, which is original work. Finally, this chapter concludes with a summary of recent work on the dimension of linear fractals.

6.1 Linear Maps and the Jacobian Matrix

Linear maps cause many of the shape deformations that begin with the letter "s" (scaling, stretching, squashing, skewing, shearing and spinning). The following definition is similar to those in many texts (such as [Foley *et al.*, 1990]).

Definition 6.1 *A map* $T : \mathbb{R}^n \to \mathbb{R}^n$ *is* linear *if and only if*

$$T(sx + y) = sT(x) + T(y) \tag{1}$$

[1] As in [Foley *et al.*, 1990], column vectors (which are popular in linear algebra) are now used, instead of row vectors, which are used here and in earlier texts.

where $x, y \in \mathbb{R}^n, s \in \mathbb{R}$ *and* $sx = (sx_1, sx_2, \ldots, sx_n)$.

Definition 6.1 has defined the two major properties of linear functions at once. The first is $w(sx) = sw(x)$, the second, $w(x + y) = w(x) + w(y)$.

Linear algebra indicates the use of matrices to specify transformations. Let M be an $n \times n$ matrix of real constants. Then xM is a linear transformation of point $x \in \mathbb{R}^n$. The Jacobian matrix of any n-dimensional map $w(x) = (w_1(x), w_2(x), \ldots, w_n(x))$ is the matrix $J(w)$ defined as

$$J_{i,j} = \frac{\partial w_i}{\partial x_j}. \tag{2}$$

The linear transformation matrix M is the Jacobian of the linear transformation $T()$, as shown in [Barr, 1984].

6.2 Affine Maps and Homogeneous Coordinates

Translation is not a linear function. Let y be a translation vector. Then, for $s \in \mathbb{R}$ in general,

$$(sx) + y \neq s(x + y). \tag{3}$$

An affine map is a linear transformation followed by a translation.

Definition 6.2 *A map* $w : \mathbb{R}^n \to \mathbb{R}^n$ *is* affine *if and only if*

$$w(x) \equiv T(x) + y \tag{4}$$

where $T()$ *is a linear map and the translation parameter* $y \in \mathbb{R}^n$ *is independent of the function variable* $x \in \mathbb{R}^n$.

The Jacobian matrix of an affine function is the same as for its linear part. Thus, the Jacobian matrix does not sufficiently describe an affine function. Matrices may still be used to uniquely describe affine functions, but we first must use homogeneous coordinates.

Definition 6.3 *The* homogeneous coordinate system *of* $\mathbb{R}^n$ *is the space* $\mathbb{R}^{n+1}$ *along with the equivalence relation*

$$(\omega x_1, \omega x_2, \ldots, \omega x_n, \omega) \equiv (\varpi x_1, \varpi x_2, \ldots, \varpi x_n, \varpi) \tag{5}$$

for all $\omega, \varpi \in \mathbb{R} \setminus \{0\}$.

A point $p = (p_1, p_2, p_3)$ is represented homogeneously as

$$\begin{aligned} \mathbf{p} &= (p_1/\omega, p_2/\omega, p_3/\omega, \omega) && (6) \\ &= (p_1, p_2, p_3, 1). && (7) \end{aligned}$$

As the homogeneous coordinate $\omega \to 0$ the other coordinates approach infinity. A homogeneous values with null ω coordinate is a "point at infinity." These infinity points still have direction, so we will use them to denote vectors. A vector $v = (v_1, v_2, v_3)$ is represented homogeneously as

$$\vec{\mathbf{v}} = (v_1, v_2, v_3, 0). \tag{8}$$

Hence, points and vectors are distinct in the homogeneous coordinate system.

Addition and subtraction of vectors acts like it should. Any number of vectors may be added to a point to create a new point; addition of two points is not defined. Points may be subtracted to create a vector, but only after they have been normalized such that their last coordinate (ω) agrees.

Homogeneously, one can specify an affine map $T + \vec{\mathbf{y}}$ by a simple extension to the Jacobian of its linear part. Let M be an $(n+1) \times (n+1)$ matrix constructed

$$M = \left[\begin{array}{c|c} J(T) & \mathbf{0} \\ \hline \vec{\mathbf{y}} & 1 \end{array}\right] \tag{9}$$

where $\mathbf{0}$ is the column vector $(0, 0, 0)^T$. Then the following is true

$$T(\mathbf{x}) + \vec{\mathbf{y}} = \mathbf{x}M. \tag{10}$$

6.3 Linear Fractals

Fractal sets created from RIFS's of contractive affine transformations are called linear fractals.

Definition 6.4 *A set $A \subset \mathbb{R}^n$ is* linear fractal *if and only if it is (locally) fractal and it is the attractor of a RIFS of affine contractions.*

This definition depends greatly on one's personal definition of "fractal." Here local fractal is used though others may define fractals differently.

6.4 Dimension Computations for Linear Fractals

The Hausdorff dimension of an attractor of an IFS of similtudes with the open set property is simply its self-similarity dimension. When the similtudes of an IFS on $\mathbb{R}$ overlap, [Falconer, 1987] shows that the Hausdorff dimension still equals the self-similarity dimension almost all the time. For a general IFS of affine transformations on $\mathbb{R}^n$, computing the Hausdorff dimension of its attractor is quite complicated.

6.4.1 The Hausdorff Dimension of IFS Attractors

Let $B = B_{r=1}(0) \subset \mathbb{R}^n$ be the n-dimensional unit ball at the origin. Then its image under a non-singular linear transformation $T(B)$ is an n-dimensional (hyper-)ellipsoid with n principle axes. The length along each principle axis from the origin to the ellipsoid's surface defines each *singular value* of T. Notice the maximum singular value is equal to the Lipschitz constant of the linear map. Thus, the singular values of a non-singular contraction all lie within the open interval $(0, 1)$.

Definition 6.5 *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be a linear contraction. Denote its chain of singular values as*

$$0 < s_1 \leq s_2 \leq \cdots \leq s_n < 1. \tag{11}$$

Then for any dimension $0 \leq d \leq n$ define the singular value function *as*

$$\phi^d(T) = s_1 s_2 \cdots s_{\lceil d \rceil - 1} s_{\lceil d \rceil}^{1+d-\lceil d \rceil} \tag{12}$$

and for $d > n$ as

$$\phi^d(T) = (s_1 s_2 \cdots s_n)^{\frac{d}{n}}. \tag{13}$$

The singular value function can then be used to define a dimension based on the affine maps of an IFS.

Definition 6.6 *Let $\{T_i + \mathbf{y}_i\}_{i=1}^N$ be an IFS of n-dimensional affine transformations. Then the following dimension is defined*

$$d(T_1, \ldots, T_N) = \inf\{d : \sum_{i=1 \ldots N} \phi^d(T_i) < \infty\}. \tag{14}$$

For lack of a name, we will term this value the Falconer dimension.

Furthermore, the following theorem shows that this dimension $d(T_1, \ldots, T_N)$ is almost always equal to the Hausdorff dimension of the attractor of the IFS when $d(T_1, \ldots, T_N)$ is less than n, the dimension of the embedding space. Otherwise the attractor's dimension is clamped at n.

Theorem 6.1 *Let A be the attractor of an IFS $\{T_i + \mathbf{y}_i\}_{i=1}^N$ of n-dimensional affine transformations such that* $\mathrm{Lip} T_i < \frac{1}{3}$. *Then for almost all $\mathbf{y}_i \in \mathbb{R}^n$ (with respect to the $(n \times N)$-dimensional Lebesgue measure)*

$$\dim A = \min\{n, d(T_1, \ldots, T_N)\}. \tag{15}$$

Proof: ... is found in [Falconer, 1988]. □

Furthermore, when the attractor of an IFS meets the criterion of the preceding theorem, then the following theorem states that computing the Hausdorff dimension of the attractor is a simple as computing its box-counting dimension.

Theorem 6.2 *Let A be the attractor of an IFS $\{T_i + \mathbf{y}_i\}_{i=1}^N$ of n-dimensional affine transformations such that*

$$\dim A = \min\{n, d(T_1, \ldots, T_N)\}. \tag{16}$$

Then

$$\dim_B A = \dim A. \tag{17}$$

Proof: ... is found in [Falconer, 1988]. □

6.4.2 The Box-Counting Dimension of RIFS Attractors.

Very little work has been done on the Hausdorff dimension of an RIFS attractor. One recent result, from [Barnsley *et al.*, 1989], is a formula for the box-counting dimension of a RIFS attractor. However, since this result is not clearly related to the Hausdorff dimension, and much different nomenclature is used in their theorem and proof, we refer the reader to [Barnsley *et al.*, 1989] for further information on this result.

Part III

MODELING

Chapter 7

MODELING WITH LINEAR FRACTALS

Often models capitalize on redundancies to make object specification simpler. Polygons are used because they concisely describe a compact planar set of points. All of the points in a polygon can be derived from a list of the polygon's vertices so polygons are commonly specified this way. Cyclical textures, such as a checkerboard, are redundant across space and so require only the modeling of a single section along with a description of how to copy it to make the rest of the pattern.

Fractals too have redundancies, though not across space but across scale. A checkerboard, extended infinitely, will cycle forever as one traverses its surface. A linear fractal will likewise cycle forever, as one zooms into its surface.

7.1 Explicit versus Implicit Models

Explicit models have enjoyed a long-standing popularity in computer graphics. Describing an object by enumerating points on its surface is a very natural way of modeling and lends itself

easily to interactive specification. Explicit models are also very flexible, directly allowing local and global alterations.

Implicit models have gained popularity in recent years. Specification of very detailed shapes is often easier when done implicitly, by describing them as loci. Such models do not require the modeler to manipulate each local part of the model. Instead they require the specification of some condition for the points to meet.

One may model the unit circle at the origin using one of two methods. The explicit function

$$f(\theta) = (\cos\theta, \sin\theta), \tag{1}$$

models the circle as the image of the interval $[0, 2\pi)$ whereas the implicit function

$$f(x, y) = x^2 + y^2 - 1 \tag{2}$$

models the circle as a zeroset. In the first example, the circle is described parametrically whereas in the second, it is described as a locus. The explicit model (1) has the advantage of point enumeration — it is easy to draw a picture of the circle by plotting the image of a finite subset of input points. The implicit model (2) has the advantage of testing — it is easy to check if a particular point is inside, on or outside the circle.

7.2 The RIFS Model

The RIFS model is both an explicit and an implicit model. Both algorithms have drawbacks: the explicit method does not work on all RIFS models whereas the implicit RIFS model requires augmentation to the RIFS digraph.

7.2.1 The Explicit RIFS Model

Treated explicitly, the RIFS models an attractor using the Chaos Game [Barnsley *et al.*, 1988]. The Chaos Game is based on a theorem that states that the fixed points of randomly

chosen maps are dense in the attractor (Theorem 3.1.3(i) in [Hutchinson, 1981]). The Chaos Games enumerates points; the RIFS here is treated as an explicit model.

Algorithm 7.1 (Chaos Game) *Let* $(\mathbf{w}, G)$ *be an RIFS with a* strongly-*connected digraph* G. *Then the attractor of the RIFS is approximated by*

1. Let $\mathbf{x}$ be any initial point and let v_i be any vertex in G_v.
2. Initialize $k = 1$ and do the following forever ...
2.1. Pick, at random, an integer $j \in 1 \ldots N$ such that $(v_i, v_j) \in G_e$.
2.2. Let $\mathbf{x} = w_j(\mathbf{x})$.
2.3. If $k > 43$ then plot $\mathbf{x}$.
2.4. Let $i = j$ and increment k.
2. End "do."

The Chaos Game generates a point cloud. This point cloud may appear more dense in places and sparser in others. This can be remedied by setting the probabilities that an n-dimensional map is randomly chosen based on the Lipschitz constant of that map taken to the nth power [Barnsley *et al.*, 1988].

As rigorously shown in [Barnsley *et al.*, 1989], the Chaos Game does not work on a RIFS with a weakly-connected digraph. This is because a point can get trapped in a section of the digraph, never reaching some of the vertices in the digraph, thereby leaving much of the RIFS attractor unplotted.

7.2.2 The Implicit RIFS Model

The RIFS model may also be treated as an implicit model, though this requires the construction of a set B that contains the RIFS attractor. If the attractor of an RIFS is unknown, then a method described in Chapter 14 automatically generates a bounding set.

A new digraph G', created by adding a single vertex v_0 to G, simplifies the implicit generation algorithm. Let C be the condensation of G. Then, for every vertex of C (corresponding to a strongly connected component of G) with in-degree of zero, add an edge to G'_e from v_0 to

every vertex in the strongly-connected component of G' corresponding to the zero in-degree vertex in C. The result is a "start vertex" v_0.

Algorithm 7.2 *Let $(\mathbf{w}, G')$ be an RIFS with attractor $A \subset \mathbb{R}^n$ and let $B_0 \subset \mathbb{R}^n$ be any non-empty set of finite diameter that contains A. Let $\mathcal{B}$ be an empty stack of tuples of the type (B, i) where $B \subset \mathbb{R}^n$ and i is an integer from 1 to N.*

1. Let $i = 0$.
2. If $x \in B_0$ then push B_0 onto $\mathcal{B}$.
3. While $\mathcal{B}$ is not empty ...
3.1. Pop $\mathcal{B}$, let (B, i) be the popped tuple.
3.2. For each $\{j : (v_i, v_j) \in G'_e\}$ ...
3.2.1. Let $B_j = w_j(B)$.
3.2.2. If $x \in B_j$ then push (B_j, j) onto stack $\mathcal{B}$.
3.2. End "For."
3. End "While."

If the preceding algorithm finishes, then $x \notin A$.

The implicit algorithm shows point exclusion is decidable but point inclusion is likely undecidable, a result similar to [Penrose, 1989].

7.3 The L-System Model

Grammar-based models have always been a popular choice for modeling linear fractals [Smith, 1984; Prusinkiewicz *et al.*, 1988]. An L-system[1] is a deterministic parallel context-free grammar. Parallel means the production rules are not applied right-to-left nor left-to-right but in parallel — all at once.

An L-system consists of an initial string and a set of production rules of the form

$$A \to B \tag{3}$$

[1]Specifically, a D0L-system.

where A is a single uppercase character and B is a string of one or more uppercase and/or lowercase characters. An uppercase character denotes a production rule whereas a lowercase character denotes a character in the final word this L-system is describing. In this context, most "words" will be infinitely long. The set of all words an L-system describes is called its *language.*

For example, the L-system described by a starting symbol A and production rules

$$A \rightarrow BcAcB \tag{4}$$

$$B \rightarrow AdBdA \tag{5}$$

produces infinitely long words characterized as the limit of the sequence

$$\begin{gathered} A, BcAcB, AdBdAcBcAcBcAdBdA, \\ BcAcBdAdBdAdBcAcBcAdBdAcBcAcBcAdBdAcBcAcBdAdBdAdBcAcB, \ldots \end{gathered} \tag{6}$$

This sequence seems organized, though this terse representation hides its true meaning.

Turtle geometry [Abelson & diSessa, 1982] is used to visualize the words in an L-system's language. In this context, a turtle is a small robot holding a pen. This turtle can perform several operations with the pen. We will restrict the turtle to the three operations. Let c and d cause the turtle to draw a fixed length straight line segment and then cause the turtle to turn 60° to the left and right, respectively. Then four elements in the sequence (6) corresponds to the graphs in Figure 7.1. Now the limit word is obvious — it is Sierpinski's gasket.

Larger character sets are usually used, with such advanced turtle actions as pushing and popping the current state of the turtle [Smith, 1984]. In [Prusinkiewicz & Lindenmayer, 1990], more sophisticated 3-D turtle operations are defined, some with optional parameters.

Of particular interest is Chapter 8 of [Prusinkiewicz & Lindenmayer, 1990], which shows how any L-system can be reduced to a RIFS. In most applications, L-system models are evolved to fixed levels, where they terminate with Euclidean primitives. For these two rea-

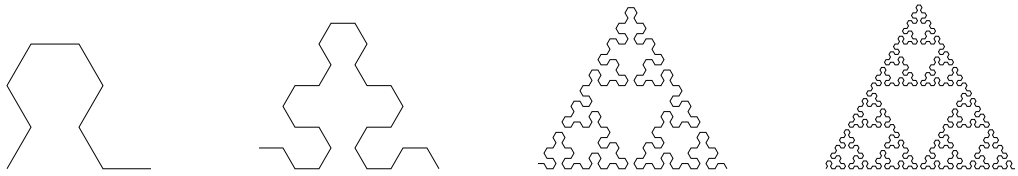

Figure 7.1: Turtle interpretations of four words generated by an L-system.

sons, this dissertation concentrates specifically on the RIFS model of linear fractals, without any loss of generality.

Chapter 8

THE COLLAGE THEOREM

The collage theorem [Barnsley *et al.*, 1986] states that if one can approximate a shape by efficiently covering it with a finite number of smaller transformed self-replicas, then the attractor of the IFS defined by these transformations provides an even closer approximation of the original shape.

The collage theorem is more than just a theoretical result on recurrent iterated function systems, it is a philosophy for modeling. When given a shape to model, the modeler ponders the object and examines its components, looking for similarities. For each component the modeler must recall, from a library of distortions, different ways the object can be deformed into it. If none perform satisfactorily, then either the object must be subdivided differently or the component itself must be subdivided into smaller pieces.

A little experience with the collage theorem helps one to see shapes in an entirely new way: a square is four smaller squares sharing a corner; a tree is a trunk with smaller trees extending from it.

The collage theorem's main drawback is the need of intuition to find the best collage. This has impeded algorithmic solutions to the inverse problem using the collage theorem. Some solutions have ventured into the field of artificial intelligence, an area many geometric

modelers seem unfamiliar with.

8.1 Statement and Proof

The collage theorem is stated here similar to its original statement in [Barnsley *et al.*, 1986]. It is actually just a simple corollary to Theorem 3.5, but its mere statement creates a powerful tool in the control of fractal shapes.

Theorem 8.1 (The Collage Theorem) *Let $B \in \mathbb{R}^n$ and let $\mathbf{w} = \{w_i\}_{i=1}^N$ be an IFS with attractor A and contractivity factor $\mathbf{s}$. Then*

$$h(A, B) \leq \frac{h(B, \mathbf{w}(B))}{1 - \mathbf{s}}. \tag{1}$$

Proof: Consider the following chain of bounds

$$\begin{aligned} h(B, \mathbf{w}^{\circ n}(B)) &\leq \sum_{i=1}^{n} h(\mathbf{w}^{\circ (i-1)}(B), \mathbf{w}^{\circ i}(B)) && (2) \\ &\leq \sum_{i=1}^{n} \mathbf{s}^{i-1} h(B, \mathbf{w}(B)) && (3) \\ &\leq \frac{1 - s^n}{1 - \mathbf{s}} h(B, \mathbf{w}(B)). && (4) \end{aligned}$$

We have the right-hand side of (2) by the triangle inequality on the Hausdorff metric; (2) implies (3) by Theorem 3.5; (3) implies (4) by some simple but tricky algebra. Taking the limit of both sides as $n \to \infty$ gives

$$h(A, B) \leq \frac{h(B, \mathbf{w}(B))}{1 - \mathbf{s}}. \tag{5}$$

since $\mathbf{w}^{\circ n}(B) \to A$ (Corollary 3.7) and $\mathbf{s}^n \to 0$ as $n \to \infty$ (since $\mathbf{s} < 1$). □

The Hausdorff distance is used to measure the error between the target set and the attractor. As this value goes to zero, the IFS attractor converges to the target set.

The quality of the IFS approximation depends on the contractivity of its maps. An IFS does such a good job at removing the error of the collage because the error in the global approximation is reduced in its smaller component approximations.

8.2 Recurrent Collages

The recurrent collage theorem, and its proof, appear originally in [Barnsley *et al.*, 1989]. Once the correct tools are in place, its statement and proof are simple extensions of the ordinary collage theorem.

Theorem 8.2 (The Recurrent Collage Theorem) *Let* $\mathbf{B} \in (\mathbb{R}^n)^N$ *and let* $(\mathbf{w} = \{w_i\}_{i=1}^N, G)$ *be a RIFS of with attractor* $\mathbf{A}$ *and contractivity factor* $\mathbf{s}$. *Then*

$$h^N(\mathbf{A}, \mathbf{B}) \leq \frac{h^N(\mathbf{B}, \mathbf{w}^N(\mathbf{B}))}{1-s}. \tag{6}$$

Proof:

$$\begin{aligned} h^N(\mathbf{B}, (\mathbf{w}^N)^{\circ n}(\mathbf{B})) &\leq \sum_{i=1}^{n} (h(\mathbf{w}^N)^{\circ(i-1)}(B), (\mathbf{w}^N)^{\circ i}(B)) && (7) \\ &\leq \sum_{i=1}^{n} \mathbf{s}^{i-1} h^N(\mathbf{B}, \mathbf{w}^N(\mathbf{B})) && (8) \\ &\leq \frac{1-s^n}{1-\mathbf{s}} h^N(\mathbf{B}, \mathbf{w}^N(\mathbf{B})). && (9) \end{aligned}$$

Taking the limit as $n \to \infty$ gives

$$h^N(\mathbf{A}, \mathbf{B}) \leq \frac{h(\mathbf{B}, \mathbf{w}^N(\mathbf{B}))}{1-\mathbf{s}} \tag{10}$$

in the same manner as in the previous proof. □

Though it is more powerful, it is also significantly more difficult to model with the recurrent collage theorem. One must not only be able to describe a set out of smaller copies, one must be able to describe a set using portions of smaller copies.

Hierarchical methods seem more intuitive. One may model a tree with an RIFS by using one IFS to generate the main branching structure, another for the leaves, and a third for the bark. These three IFS models may then be incorporated hierarchically into a single weakly-connected RIFS model. The multifractal observations of many natural phenomena [Shroeder, 1991] seem to imply the hierarchically-constructed RIFS models will be a useful tool in modeling nature.

Chapter 9

INTERACTIVE MODELING

As mentioned in the last chapter, modeling with the Collage Theorem is usually an interactive process. This chapter discusses three different methods implemented by the author to model a linear fractal by interactively specifying the affine maps of an iterated function system.

The first method was to compose the IFS by a textual modeling language developed by the author. Here the results were realistically rendered (by an algorithm described later) so the interactive response time was lengthy, typically five minutes.

The other methods composed the affine maps of an IFS interactively in real time. One method was to specify the parameters of a particular affine transformation interactively with a mouse. The other was to apply free-form linear deformations to an object to specify an affine transformation.

An informal study was performed on these latter two methods as an exercise in computer-human interaction. Its results were not conclusive enough to warrant any one technique over the other. They were useful, however, in fine-tuning these systems.

9.1 Textual Specification by Map Composition

One direct method for describing an affine map is to specify its components, such as scaling in a direction, rotation about an axis and translation to a new location. An efficient exact way to get this information is to make the user type in the commands directly. As expected, this technique leaves a lot to intuition.

As found in many modeling languages, a current homogeneous affine transformation matrix was maintained in constructing each map of the IFS. The current matrix was initialized to the unit 4×4 matrix, corresponding to a null affine transformation.

The following forms of affine transformations were allowed:

- **Scale:** x, y, z. Component scaling, independently for each axis.

- **Rotate:** axis, θ. Rotation by θ degrees about an axis according to the left-handed coordinate system.

- **Translate:** x, y, z. Translation along a vector.

- **Matrix:** $m_{1,1}, \ldots, m_{1,3}, m_{2,1}, \ldots, m_{4,3}$. A catch-all, handling skews, shears or any other strange affine transformation.

This method was used in a slow interactive visualization loop where the affine maps were composed in a modeling file and a small picture of the resulting attractor was rendered in a few minutes. Most of the actual modeling performed in this manner was done in the user's head or taken from a book on the user's shelf. The most successful application of this method was to fine tune the IFS model, checking its final results after each change of the IFS parameters.

9.2 Graphical Specification by Map Composition

To speed up the interactive visualization loop, one can display a sample of the map visually as a "before and after" set. A tetrahedron was used to show the results of an affine map[1].

An ambiguity arises when the tetrahedron is positioned such that one of its "after" vertices is closer to another "before" vertex than the original "before" vertex. This is fixed by one of two solutions. One solution is to connect before and after vertices by a specially indicated line. The other is to indicate each of the four vertices of a tetrahedron uniquely, say by assigning them four distinct hues.

To aid in interactivity, each canonical transformation can be entered through a continuous input device, such as a mouse. Then the resulting transformation can be displayed in real time while the user adjusts it to meet some criterion.

The attractor of the IFS was displayed immediately after each complete transformation adjustment. The attractor was generated using the Chaos Game (Algorithm 7.1) and displayed in real time as a point cloud. This point cloud could rotate if desired to aid in perceiving it in 3-D.

9.3 Graphical Specification by Shape Transformation

Alternatively, one may specify an affine map in $\mathbb{R}^n$ by specifying two sets of $n+1$ points $X = \{x_1, x_2, \ldots, x_{n+1}\}$ and $Y = \{y_1, y_2, \ldots, y_{n+1}\}$. In 3-D, X and Y are tetrahedra.

Using the mouse, the modeler grabs a vertex of a tetrahedron, moves it parallel to the image plane in the viewing window the mouse is currently in, and places the vertex in its new location. Then the vertices of the two tetrahedra are used to evaluate the affine transformation from the master tetrahedron to the instance tetrahedron.

[1]This technique is similar to much of the software currently available to design 2-D iterated function systems. The author is unaware of any such previously written software for 3-D iterated function systems.

This affine transformation matrix M is found by solving the linear system

$$x_i M = y_i \tag{1}$$

for all $x_i \in X$ and $y_i \in Y$. This produces three systems of four linear equations with four unknowns which can be solved symbolically by any number of current math packages. The resulting equations are then used, in real time, to dynamically generate the affine transformation as the tetrahedra are modified.

Chapter 10

ALGORITHMIC MODELING

Even the simplest kinds of shape transformations require several parameters to be fully described. Hence, the parameter space of iterated function systems is quite large, since every parameter of every transformation adds a new dimension to this space. Searching this parameter space exhaustively is much too time consuming to be useful.

Instead, more efficient methods have been developed to find the parameters of a linear fractal model that specifies a desired set. The rest of this chapter will concentrate on using the Method of Moments, a popular parameter solving method.

The methods described hereafter originated in [Barnsley & Demko, 1985] but have been explored much more vigorously in [Vrscay & Roehrig, 1989; Vrscay, 1991].

10.1 The Method of Moments

The Method of Moments requires a large number of moments to accurately determine the proper parameters. The power moments of a set offer a countably large number of moments for which to solve.

Definition 10.1 *Let $A \subset \mathbb{R}^n$. Then the* power moments $g_{i_1,i_2,\ldots,i_n}$ *of A are defined as*

$$g_{i_1,i_2,\ldots,i_n} = \int_A x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} dA. \tag{1}$$

The moment $g_{0,0,\ldots,0} = \int_A dA = 1$ for all A.

Given two sets A and B, with moments g, h, it is not too difficult to see that if $g_{i_1,i_2,\ldots,i_n} = h_{i_1,i_2,\ldots,i_n}$ for all sequences $i_1, i_2, \ldots, i_n$ then $A = B$. If a large number of moments match between two sets, then they are likely similar, though there is no guarantee of this.

The Method of Moments estimates parameters of a model by computing the moment of the model's result explicitly, deriving the equations for the moments of the model's results in terms of its parameters, and then inverting these equations to solve for the parameters in terms of moments.

10.1.1 A Parameterized Iterated Function System

The rest of this chapter will deal with an IFS parameterized as

$$w_i(x) = s_i x + a_i \tag{2}$$

where s_i, a_i and x are points in either $\mathbb{R}$ or $\mathbb{C}$.

The moments of such an IFS[1] are produced recursively by the recurrence equation

$$g_m = \frac{\sum_{j=1}^{m} \binom{m}{j} g_{m-j} \frac{1}{N} \sum_{i=1}^{N} s_i^{m-j} a_i^j}{1 - \frac{1}{N} \sum_{i=1}^{N} s_i^m}. \tag{3}$$

10.1.2 Matching Moments Analytically

Section 3.3 of [Barnsley & Demko, 1985] solves for the complex IFS parameters s_i, a_i for the twindragon. Their method is analytic, and they assume *a priori* that the IFS contains two maps, $s_1 = s_2$, and $a_1 = -a_2$.

[1] Assuming a uniform measure.

As prescribed by the Method of Moments, the moments of the twindragon are computed explicitly by summing x^n over the number of pixels the twindragon covers.

The moment $g_0 = 1$, by definition. The odd moments $g_1, g_3, \ldots$ vanish since the twindragon is symmetric about the origin. Thus, one needs two moments, say g_2 and g_4, to solve for the two parameters s_1 and a_1.

They obtain analytically, from (3), the solution $s \approx 0.064 - 0.594i$. Though it differs slightly from the actual parameter $s = i/2$, the approximated attractor barely resembles the original (compare Figures 11 and 12 in [Barnsley & Demko, 1985]). Better approximations can be made by explicitly computing the moments on a finer grid.

10.1.3 Matching Moments Procedurally

The previous example matched moments analytically by computing inverse equations. When the number of parameters increases, by allowing a larger number of contractions of a more general type, the moment equations increase in degree, requiring root-finding methods to solve for the parameters.

Let $(\mathbb{R}^2)^N$ be the parameter space of tuples

$$\mathbf{p} = ((s_1, a_1), \ldots, (s_N, a_N)). \tag{4}$$

The norm (magnitude) $|\mathbf{p}|$ of a tuple $\mathbf{p} \in (\mathbb{R}^2)^N$ is given by

$$|\mathbf{p}| = \sum_{i=1}^{M} g_i(\mathbf{p} - G_i) \tag{5}$$

where M is the number of moments considered and the G_i are the explicitly measured moments of the target set. As $M \to \infty$ the norm of the parameters that create the target set will necessarily vanish. Furthermore, the norm is smooth with respect to the parameters. Specifically, its gradient may be computed in closed form.

Thus [Vrscay & Roehrig, 1989] proposes a gradient search for parameters of zero norm. The unfortunate drawback to gradient searches, and other local searches, are that they get

stuck at local minima. Improvements such as simulated annealing and genetic algorithms appear to be better suited for such searches, the latter shown to be quite effective for the inverse problem [Cabrelli *et al.*, 1991].

10.2 Block Coding

No review of automatic IFS generation would be complete without mention of its applications to image compression. Perhaps the most successful implementation of such an algorithm is [Jaquin, 1991]. This method, a fractal block-coding scheme, does not operate on sets of points in $\mathbb{R}^2$. Instead, it operates on blocks of images. As such, the methods described in this dissertation are unrelated to this method. Those interested in this application of iterated function systems should read [Jaquin, 1991].

Part IV

RENDERING

Chapter 11

LINEAR FRACTALS: WHERE CLASSICAL RENDERING FAILS

Fractals are difficult shapes to render; their infinite detail hampers occlusion computation, lacks surface normals and aliases at all sampling resolutions. This chapter outlines current rendering techniques and demonstrates their shortcomings regarding linear fractals. The following chapters contain the main contributions of this dissertation: overcoming the problems of rendering linear fractals.

11.1 Occlusion and the Infinitely Detailed

Occlusion is, by far, the most powerful 3-D cue and is the basis of most modern rendering methods. Occlusion computations for fractals are exceptionally difficult due to the extreme detail of their surfaces. The difficulty of determining fractal occlusion is perhaps the main reason so many have limited themselves to 2-D for investigating fractal sets.

11.1.1 The Hidden-Surface Problem

The hidden-surface problem (determining occlusion) is a classic problem whose solution is fundamental in 3-D computer graphics. A large number of solutions to this problem are summarized in [Sutherland *et al.*, 1974], where they were categorized as object-space or image-space[1] algorithms.

Object-space methods determine hidden surfaces to machine resolution whereas image-space methods determine hidden surfaces to screen resolution. The infinite detail of a fractal surface will likely contain an extreme number of occlusions, suggesting the use of image-space algorithms [Hart, 1989].

Two image-space methods for removing hidden surfaces are z-buffering and ray casting. The z-buffer method has been used in other fractal rendering algorithms and is described here for completeness. The hidden surface method used in the fractal rendering algorithms described herein will be ray casting.

Z-buffering

One uses a z-buffer to eliminate hidden surfaces by plotting every primitive of a scene. Each pixel maintains a minimum distance (z-value). When a primitive projects onto the pixel, the distance from the eye to the point on the primitive that maps to the pixel is measured. If and only if this distance is less than the minimum, then the minimum is updated and the pixel receives the primitive's color [Catmull, 1975].

A z-buffer method solved the hidden-surface problem for fractals in [Norton, 1982; Norton, 1989b]. Although the z-buffer method is image-space, it is object-time — it requires a complete pass through all of the primitives in an object database.

As the resolution increases, such as in a close-up, the computation time increases dra-

[1] a third "list-priority" category was used for algorithms with elements of both image- and object-space algorithms.

matically. This was experienced in the creation of [Norton & Melton, 1988], where it was overcome by improving the resolution locally, *a priori* — only in the areas they knew would be closely inspected.

Ray Casting

Rays in ray casting originate at the eye point and pass through the centers of pixels on the image plane. For each pixel's ray, the object that intersects that ray closest to its origin is the object visible through that pixel [Appel, 1968].

Ray casting is an image-space algorithm since the number of rays is proportional to the image resolution. Ray casting is also an image-time algorithm in that its computation time depends more on image resolution than on object database size. Hence, when rendering close-ups of fractal sets, the amount of computation time should not increase as dramatically as it would using object-time methods.

Ray tracing extends ray casting by casting secondary rays at the point of intersection [Whitted, 1980]. Rays cast from the intersection point to the light source determine the shadows cast upon the intersection point. Rays cast off the object as a ricochet of the original ray determine the image a shiny surface reflects. Rays cast through the object simulate transparency, and when their direction is properly adjusted, they exhibit refraction effects. In short, ray tracing provides a simple model for illumination.

11.1.2 Ray-Fractal Intersection

Ray tracing depends greatly on the ability to efficiently compute the intersection of a ray with an object. The efficiency of a ray-object intersection computation depends largely on the internal representation of the object. Linear fractals may be represented by many standard computer graphics models. One, in particular, permits an efficient ray intersection.

Linear Fractals as CSG Models

Commonly, the first intersection of a ray with an object is determined from a finite set of ray-object intersections. For example, when evaluating the intersection of a ray with a CSG constructed object[2], one computes the intersection of the ray with every primitive in the construction, then merges these intersections together via a Roth diagram [Roth, 1982].

When the object is fractal, one must find the first of a continuum of discrete intersections with the object. When formulated this way, ray intersection is intractable.

Linear Fractals as Implicit Surfaces

Another method for determining ray intersection is the use of iterative methods which converge on the ray-object intersection closest to the ray origin. Such methods have been developed and successfully used, in general, to ray trace general arbitrary implicit and explicit surfaces [Kajiya, 1982; Hanrahan, 1983; Toth, 1985; Kalra & Barr, 1989].

These techniques rely on root finding methods, such as Newton's method, which rely on the derivative of the function that defines the object. When this function defines a fractal surface, its derivative is no longer defined and these methods fail.

If the implicit function underestimates the distance from any point to the object, then one can create a sequence that converges to the first ray-object intersection [Hart *et al.*, 1989]. Such a distance function was devised for linear fractals in [Hepting *et al.*, 1990; Hepting, 1991]. Thus, linear fractals may be rendered using this distance estimate, though not in optimal time.

Linear Fractals as Recursive Hierarchies

A more efficient method for ray intersection is developed in Chapter 12. There, linear fractals are defined as the natural result of a cyclic object hierarchy. A few minor enhancements

[2] An object constructed using union, intersection and complement operations on other objects

augment the standard ray-hierarchy intersection method, creating an efficient ray-fractal intersection algorithm.

11.1.3 Volume Rendering and Geometric Measure Theory

Iterative methods produce approximate solutions, since one cannot wait forever for an infinite series to converge. This error is mandatory if certain sets are to be rendered at all, as pointed out in the next example.

A paper on rendering fur [Kajiya & Kay, 1989] introduced the delicate relationship between volume rendering[3] and geometric measure theory:

> ... consider the rendering of a single plane surface via a volume density. Assume the surface is stored into a volume density so that it bisects the cube. The optical depth of the surface is so high that it simulates an opaque surface. ... For the transparency calculation, even though the optical depth parameter is set very high, the line integral of the density in the exponent will be vanishingly small. This is because the surface is infinitely thin, the the line integral will pierce the surface at only a single point. This yields an integral of 0. A similar problem occurs in the brightness calculation. ... Thus the transparency and brightness for this surface will both be zero — an invisible surface!

One expects such a problem when measuring a 2-D set with a 3-D measure. This problem arises with any set of dimension less than three.

One solution to this problem is to thicken the set[4], making it measurable in 3-D. If A is

[3]It is important to understand the context of the following quote which describes a shortcoming of modeling an object as an array of volume densities. Volume rendering is definitely not limited to this paradigm as [Kajiya & Kay, 1989] quite clearly demonstrates.

[4]As noted by D. Sandin, proper treatment of the wave properties of light makes such immeasurable objects visible. The wavelengths of light thicken the rays of light, which is a viable alternative to thickening the set.

a set to be rendered, then render the set $A + \epsilon$.

The next step is to determine the magnitude of ϵ. If ϵ is too large then the rendering may incorrectly portray the topological properties of the set; it may appear heavier or more connected than it really is. Conversely, if ϵ is too small then the rendering might appear dust-like, again incorrectly portraying connectedness. In Chapter 15, we determine the best ϵ for properly portraying topological properties at a given resolution.

11.2 Shading Without Surface Normals

Recalling calculus, the derivative of a function $f : \mathbb{R} \to \mathbb{R}$ is defined

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{1}$$

Functions that produce fractal curves lack a derivative because a limiting sequence of slopes does not converge. Rather, this sequence becomes periodic as h decreases since fractals are symmetric across scale. The tangent of a fractal surface is similarly undefined, though several approximations have been devised.

11.2.1 Surface Normal Approximations

In [Norton, 1982], local tangent planes of points on the surface of a fractal were approximated by the gradient of the z-buffer. Though its results were quite effective, as seen in the illustrations in [Norton, 1982; Norton, 1989b] and the animations [Norton & Melton, 1988; Norton, 1989a], this method is inappropriate for ray tracing, which requires no z-buffer.

If the surface is generated to a fixed resolution, at which point it is approximated by primitives, then the surface normal of the primitive used to approximate the surface may be used. This method was used by the fractal terrain papers [Kajiya, 1983; Bouville, 1985]. When the fractals are constructed to high levels of detail, this method will likely cause aliasing.

In [Hart *et al.*, 1989], the gradient of a distance estimate function provided a decent approximate surface normal. Such distance estimates exist for linear fractals [Hepting *et al.*, 1990; Hepting, 1991] but are relatively expensive to compute.

11.2.2 Shading Module Approximations

A shading module is a procedure, rather than parameters, associated with an object that computes the shading of its surfaces [Cook *et al.*, 1984]. Shading modules are more powerful and flexible than simple shading parameters, and are used to allow novel surface shading effects in a standard rendering package.

The extra power and flexibility of shading modules is quite useful when rendering highly detailed objects. A single surface normal inadequately represents the orientation of a highly detailed surface. Instead, shading modules better portray tiny details.

Both [Kajiya & Kay, 1989] and [Thompson, 1991] identify and offer solutions to the problem of rendering highly detailed objects. Both solutions approximate complex intricate geometries with a single primitive.

Texels

In [Kajiya & Kay, 1989], highly detailed objects are modeled volumetrically. The standard volume model was enhanced by replacing the simplistic density model [Kajiya, 1983] with a sophisticated shading module at each voxel. Such improved voxels are called "texels." This shading module approximates some highly detailed sub-voxel geometry, the rendering of which is precomputed and now represented by the voxel.

Amalgams

In [Thompson, 1991], shading modules are associated with bounding volumes. Such improved bounding volumes are called "amalgams." These shading modules approximate their

contents by precomputing the reflected light of their sub-geometries in various discrete directions.

Hierarchical Shading

Chapter 17 shades linear fractal surfaces by partially illuminating its bounding volume hierarchy. It is not intended to produce an accurate depiction of the shading of a fractal surface. Instead, this kind of shading aids in the perception of surface orientation while inhibiting aliases.

11.3 Sampling Infinite Frequencies

The silhouette of a fractal surface is often a fractal itself [Falconer, 1985]. When rendering such a surface, one must remove both aliases from shading artifacts as well as those from rasterization. Hierarchical shading (Chapter 17) inhibits shading aliases. Rasterization aliases, on the other hand, must be reduced by other means.

11.3.1 The Rasterization Integral

We adapt a rasterization integral from [Norton *et al.*, 1982] for use in the ray-tracing paradigm. The rasterization integral over arbitrarily shaped pixel P, assuming a uniform measure, is

$$\int_P I(R(p))dp \tag{2}$$

where $R(p)$ is a ray extending from the eye through point $p \in P$ in the image plane and $I(R(p))$ is the resulting intensity found by tracing ray $R(p)$. Rasterization is just a specific form of integration.

11.3.2 Integration

When one reconstructs a signal from information gathered by sampling a frequency too infrequently, the result is a completely different signal than the original. This problem is called "aliasing." Aliasing due to improper rasterization appears as staircasing and moirés in still images, and as crawling and twinkling in animations. It results when the integral (2) is not approximated adequately.

Point Sampling Techniques

One method of integration is the Rectangular Rule, where samples of the function are taken at fixed intervals within the domain of integration. A box is created for each sample of width equal to the distance to the next sample and height equal to the function value at the sample point in the domain. The integral is approximated by the sum of the areas of these boxes.

Adaptive sampling is a method where neighboring samples that produce drastically differing function values are refined by taking additional samples between them [Whitted, 1980].

Stochastic sampling — a "Monte Carlo" method of integration — approximates the integral using randomly placed samples. This kind of sampling is just as prone to alias as over sampling. The advantage of stochastic sampling is the aliases are less perceptible to the human eye. Displacing samples from an initial grid in random directions is called "jittering," and causes aliases to appear as noise [Cook, 1986]. Constraining random samples to be at least some minimum distance from each other produces a Poisson-disk distribution, which mimics the organization of receptors in mammalian retinas, and improves on the perception of aliases as unstructured noise [Mitchell, 1987].

By the Sampling Theorem, the number of samples required to accurately measure a signal must be at least twice the highest frequency of the signal (the Nyquist limit) [Shannon, 1949]. When integral (2) is approximated using the Rectangular Rule, the number of samples should be at least twice the highest frequency of the function.

The infinite detail of linear fractals insures that the Nyquist limit cannot be reached by finitely many point samples. Hence, all of the above point sampling techniques are prone to alias on fractal images.

As more samples are taken, aliases are pushed into higher frequencies, where they can be eliminated by a low pass filter [Mitchell, 1987]. The main drawback to such over-sampling schemes in ray tracing is the high cost of multiple ray intersection computations.

11.3.3 Area Sampling Techniques

Area samples are efficient, requiring one sample per pixel. This one sample, however, will be more difficult to compute than a simple point sample. Taking area samples defeats the Nyquist limit problem for linear fractals since an area sample represents a continuum of point samples.

In [Catmull, 1978], the integral (2) was solved analytically by measuring the area of the projection of polygons onto the portion of the image plane bounded by a pixel. In [Heckbert & Hanrahan, 1984], this method was used to create a hybrid ray-tracing/scan-conversion algorithm called beam tracing. This method capitalizes on projection properties of polygons and is not well suited for rasterizing linear fractals.

One can approximate an area sample on the image plane by intersecting a cone with the object database instead of a ray. By thickening the ray into a cone that passes through a majority of the pixel area, the cone's intersections with objects will better approximate their projections onto the image plane [Amanatides, 1984].

The amount of the cone's visibility that an object obscures determines the percentage that object contributes to its pixel's illumination. The main drawback of "cone tracing" is the high computational expense of intersecting a cone with an object database. This high computational expense limits cone tracing to simple geometric primitives, preventing it from being an effective antialiasing mechanism for rasterizing linear fractals.

An alternative to thickening the ray is thickening the objects. The method of "covers" thickens the object instead of the ray [Thomas *et al.*, 1989]. It is described in detail in Chapter 18. Like cone tracing, the covers method also limited to simple geometric primitives. However, Chapter 18 derives a new method for the antialiased rasterization of linear fractals, a variation on the covers idea called "local covers."

Chapter 12

RAY-LINEAR FRACTAL INTERSECTION

As mentioned in the last chapter, ray tracing algorithms depend greatly on the accuracy and efficiency of ray intersection computations. In this chapter, an efficient ray-linear fractal intersection method is developed that approximates the intersection point to an accuracy sufficient for rasterization.

12.1 Definitions

We begin by defining some basic tools of ray tracing.

12.1.1 Heaps

First, a *heap* is a tree organized such that each node's index is smaller than the indices of its children. Hence, the node with minimal index is found at the top of the heap.

Definition 12.1 *A weakly-connected digraph H of weighted vertices is a heap if and only if it contains no undirected cycles (it is a tree) and any edge from a vertex v_1 to a vertex v_2*

implies $w(v_1) > w(v_2)$.

A heap is used to efficiently find the closest intersection (with respect to the ray origin) of a ray with a continually updated set of objects.

12.1.2 Rays

A ray is one-half of a line. It is anchored at an origin and extends infinitely in a single direction.

Definition 12.2 *A* ray $R(\mathbf{o}, \vec{\mathbf{d}})$ *with homogeneous origin* $\mathbf{o}$ *and direction* $\vec{\mathbf{d}}$ *is defined*

$$R(\mathbf{o}, \vec{\mathbf{d}}) = \{\mathbf{o} + t\vec{\mathbf{d}}, \forall t \geq 0\}. \tag{1}$$

Rays are parameterized by the value t. A ray intersection point may be recorded by the parameter t where the intersection happens. When the direction component $\vec{\mathbf{d}}$ is of unit length, t is the distance along the ray. The parameter of the first ray intersection is given by the following definition.

Definition 12.3 *The* first ray intersection distance t_0 *of ray* $R(\mathbf{o}, \vec{\mathbf{d}})$ *with set* B *is defined*

$$t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B) = \inf\{t : \mathbf{o} + t\vec{\mathbf{d}} \in B, t > 0\}. \tag{2}$$

The function $t_0()$ is always positive. This is so secondary rays (used for shadowing, reflection and refraction) do not immediately intersect the surface they are cast from.

12.2 Bounding Volumes and Hierarchies

This chapter builds upon the methods of [Rubin & Whitted, 1980]. These methods efficiently compute the intersection of a ray with a database of objects by a divide-and-conquer method.

The object database is partitioned, using simple geometric primitives, called "bounding volumes," to delineate spatially distinct regions.

Spheres are common bounding volumes; their intersection with rays are readily computable. Oriented bounding boxes easily bound sets given their coordinate extremes, with the added benefit of efficient ray-intersection computations. Some bounding volumes contain smaller bounding volumes, creating a hierarchy.

Such bounding volume hierarchies can be specified by digraphs. Each vertex in the digraph corresponds to an element in the hierarchy whereas a directed edge, say from vertex v_1 to v_2, implies that v_2 is directly beneath v_1 in the hierarchy. Let the vertices denote bounding volumes and edges denote a containment relation.

A low-level approximation to Sierpinski's gasket can be represented by the tree and resulting image shown in Figure 12.1.

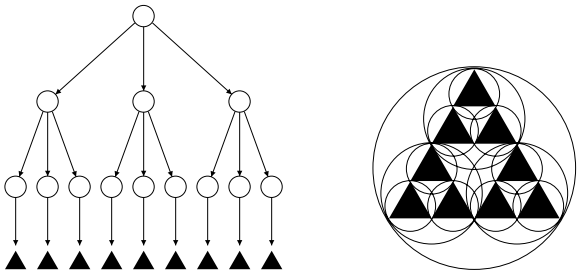

Figure 12.1: A tree topology (left) for the bounding volume hierarchy of a low-level approximation of Sierpinski's gasket (right).

Each vertex in Fig. 12.1 (left) represents the absolute position, orientation and size of a bounding volume or a primitive. The edges organize these elements into a hierarchy.

12.3 Ray-Hierarchy Intersection

Here ray intersection parses the bounding volume tree in depth first order, considering closer intersections before farther ones.

Algorithm 12.3 *Let $\mathcal{B}$ be a set of bounding volume hierarchies and primitives. Let $R(\mathbf{o}, \mathbf{b})$ be a ray and H be an empty heap. Then the first intersection with $R(\mathbf{o}, \vec{\mathbf{d}})$ and a primitive in $\mathcal{B}$ is computed by the following steps:*

1. For each bounding volume or primitive $B \in \mathcal{B}$ ...
1.1. If $R(\mathbf{o}, \vec{\mathbf{d}})$ intersects B then add $(t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B), B)$ to heap H.
1. End "For."
2. While heap H is not empty ...
2.1. Remove the minimal tuple (t_0, B) from the top of heap H.
2.2. If B is a primitive then return t_0.
2.3. Otherwise for each object B_i beneath B in the hierarchy ...
2.3.1. If $R(\mathbf{o}, \vec{\mathbf{d}})$ intersects B_i then add $(t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B_i), B_i)$ to heap H.
2.3. End "For."
2. End "While."

Well organized trees contain fewer bounding volumes than primitives. Ordinarily, the number of ray intersection computations nominally exceeds the logarithm of the number of primitives, though in the extreme case, it can be as much as twice the number of primitives.

12.4 Object Instancing

An instance is an affinely distorted copy of an object. Instancing saves space since only one "master" object database need be stored, with a 4×4 matrix representing each instance of it. Furthermore, by building instances of instances, large "fields" of similar objects may be created where the actual number of instances is only a logarithm of the number of objects appearing in the scene.

Object instancing was first described in [Sutherland, 1963] where it was used with hierarchical display lists.

Object instancing was first adapted for ray tracing in [Rubin & Whitted, 1980] as a method of reducing the size of object databases. Their technique allowed the city of Pittsburgh (38,000 primitives) to be rendered efficiently using little memory by storing only 600 actual primitives.

Procedurally generated parallelepipeds were used as bounding volumes in [Kay & Kajiya, 1986]. They exhibited a forest of trees (over 110,000 primitives) surrounding a cement pond.

In [Snyder & Barr, 1987], oriented boxes bounded objects. The authors traded space for time by using a 3-D grids structure to reduce ray intersection computations. Using these techniques they rendered a carpet (125,000 primitives), a forest (2 billion primitives) and the still unsurpassed "field of grass" (over 400 billion primitives).

Object instancing removes redundant vertices from the hierarchy by allowing bounding volumes and primitives to have more than one parent. The resulting hierarchy is no longer a tree; it is an acyclic digraph. Figure 12.2 shows the approximation of Sierpinski's gasket specified by the positioning, orientation and size of only seven bounding volumes and one primitive. Here, only the top vertex specifies a bounding volume in absolute coordinates, the other vertices each specify only a bounding volume's or primitive's relative change in placement and shape with respect to its parent in the hierarchy.

12.5 Ray-Instance Intersection

Each instance of a master object is produced by applying an affine transformation to each of its points. This can be quite tedious and memory consumptive. Each of the vertices of an instanced polyhedron must be computed from the master polyhedron, and stored for ray intersection. An sphere instanced as an ellipsoid requires a costly ray-quadric intersection computation.

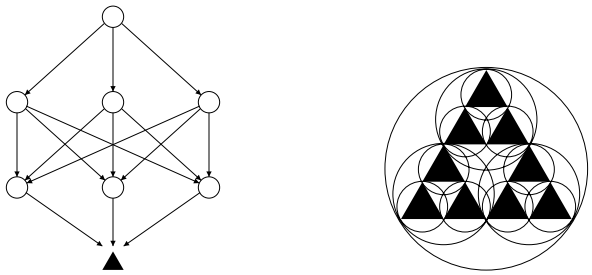

Figure 12.2: An object-instancing topology (left) for the bounding volume hierarchy of a low-level approximation of Sierpinski's gasket (right).

It is more efficient to apply the inverse affine transformation to the ray. The affine image of a ray consists of one point and one direction. Let M be an affine transformation matrix that takes master object B_0 to B_1. Then the intersection $t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B_1)$ is found by intersecting the ray

$$R(\mathbf{o}_1, \vec{\mathbf{d}}_1) = R(\mathbf{o}M^{-1}, \vec{\mathbf{d}}M^{-1}) \tag{3}$$

with the master object B_0. Note the direction component $\vec{\mathbf{d}}_1$ is probably not a unit vector. Nevertheless,

$$\mathbf{o} + t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B_1)\vec{\mathbf{d}} = M(\mathbf{o}_1 + t_0(R(\mathbf{o}_1, \vec{\mathbf{d}}_1), B)\vec{\mathbf{d}}_1) \tag{4}$$

where $B_1 = \{\mathbf{x}M : \mathbf{x} \in B\}$. In particular,

$$t_0(R(\mathbf{o}_1, \vec{\mathbf{d}}_1), B) = t_0(R(\mathbf{o}, \vec{\mathbf{d}}), MB). \tag{5}$$

Thus the t_0 values one gets from algebraic ray intersection are comparable, regardless of the instancing transformations.

12.6 Cyclic Hierarchies: A Model for Linear Fractals

Of particular interest in [Rubin & Whitted, 1980] was their treatment of bi-parametric surfaces. Using convex hull and subdivision properties, they were able to procedurally create a hierarchy of bounding boxes during ray intersection. At a fixed terminal level of the hierarchy, these bounding boxes were treated as "point" primitives.

Object instancing of fractal objects was suggested in [Fournier *et al.*, 1982] though it was first used in [Kajiya, 1983]. There procedural "cheesecake extents" (extruded triangles) hierarchically bounded a fractal mountain. A later improvement used bounding ellipsoids [Bouville, 1985]. One ray-traced fractal mountain mesh containing 262,144 primitives was shown in [Kajiya, 1983].

Most recently, object instancing was used in an animation of a multitude of robots cycling along a plane-filling curve [Amanatides & Mitchell, 1989]. Also, D. Hepting and D. Mitchell have modeled linear fractal shapes using tiny spheres but their ray-tracing programs (C. Kolb's "rayshade" and D. Mitchell's "FX"), though optimal for many other shapes, limited the renderable resolution of these linear fractal models.

The main restriction to object instancing is that the hierarchy cannot contain a loop. If a loop occurs, then some rays will intersect bounding volumes forever, never reaching a primitive.

Notice that the graph in Fig. 12.3 has no primitives. The primitives for Sierpinski's gasket are points, the limit of the bounding volume hierarchy as it converges to an uncountably infinite set of bounding volumes of zero diameter. Hence, the primitives of a cyclic digraph are the bounding volumes themselves.

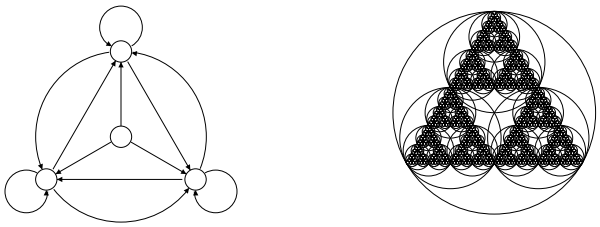

Figure 12.3: A cyclic topology (left) for the bounding volume hierarchy of Sierpinski's gasket (right).

12.7 Ray-Linear Fractal Intersection

We can use the ray-instance algorithm to intersect a ray with a linear fractal. The only necessary alteration is the bounding volume hierarchy must be terminated at some finite level, resulting in tractable ray intersection.

Algorithm 12.4 *Let each element of $\mathcal{B}$ consist of an RIFS $(\mathbf{w}, G)$, a bounding volume B such that B contains the attractor of the RIFS. Let $R(\mathbf{o}, \mathbf{b})$ be a ray and H be an empty heap. Then the first intersection with $R(\mathbf{o}, \vec{\mathbf{d}})$ and an attractor from $\mathcal{B}$ is approximated by the following steps:*

1. For each RIFS $(\mathbf{w}, G)\ldots$
1.1. Let $\mathbf{o}_1 = \mathbf{o}M^{-1}$ and $\vec{\mathbf{d}}_1 = \vec{\mathbf{d}}M^{-1}$.
1.2. If $R(\mathbf{o}_1, \vec{\mathbf{d}}_1)$ intersects B then
add $(t_0(R(\mathbf{o}_1, \vec{\mathbf{d}}_1), B), B, M^{-1})$ to heap H.
1. End "For."
2. While heap H is not empty ...
2.1. Remove the minimal triple (t_0, B, M^{-1}) from the top of heap H.
2.2. If B is a primitive then return t_0.
2.3. Or if $\mathrm{diam}(M(B)) \approx p(t_0)$ then return t_0.

2.4. Otherwise for each valid affine contraction $w_i \in \mathbf{w}$ ...
2.4.1. Let W_i be the homogeneous transform matrix representing the affine contraction w_i.
2.4.2. Let $M_i^{-1} = M^{-1} W_i^{-1}$
2.4.3. Let $\mathbf{o}_i = \mathbf{o} M_i^{-1}$ and $\vec{\mathbf{d}}_i = \vec{\mathbf{d}} M_i^{-1}$.
2.4.4. If $R(\mathbf{o}_i, \vec{\mathbf{d}}_i)$ intersects B then add $(t_0(R(\mathbf{o}, \vec{\mathbf{d}}), B), B, M_i^{-1})$ to heap H.
2.4. End "For."
2. End "While."

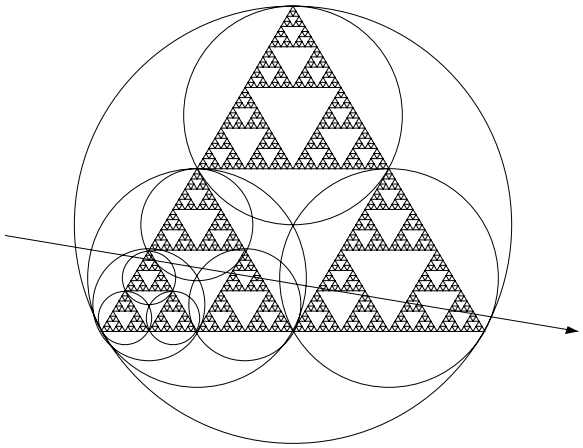

Figure 12.4: Procedural bounding volumes instanced during ray-fractal intersection.

In Figure 12.4 one ray and many instanced bounding volumes are shown. In reality, there is only one bounding volume and many instanced rays.

The algorithm finishes when a primitive — a sufficiently small bounding volume — is intersected. Always subdividing the closest intersecting bounding volume guarantees that the first intersection found is the closest to the ray origin.

12.7.1 The Bounding Volume Theorem

Algorithm 12.4 depends greatly on the bounding volume hierarchy created by the RIFS. The Bounding Volume Theorem asserts the validity and effectiveness of this hierarchy.

Theorem 12.1 (Bounding Volume Theorem) *Let* $(\mathbf{w}, G)$ *be an RIFS with invariant set*

$$\mathbf{A} = (A_1, A_2, \ldots, A_N) = \mathbf{w}^N(\mathbf{A}) \tag{6}$$

and let $\mathbf{B} \supset \mathbf{A}$ *be a bounding set* N*-tuple of* $\mathbf{A}$*. Let*

$$\mathbf{B}* = (B_1*, B_2*, \ldots, B_N*) = \mathbf{w}^N(\mathbf{B}) \tag{7}$$

denote the image of $\mathbf{B}$ *under the recurrent Hutchinson operator. Then* $A \subset B*$ *and*

$$h^N(\mathbf{A}, \mathbf{B}*) \leq \mathbf{s}h^N(\mathbf{A}, \mathbf{B}). \tag{8}$$

Proof: The premise $\mathbf{A} \subset \mathbf{B}$ implies

$$\mathbf{w}^N(\mathbf{A}) \subset \mathbf{w}^N(\mathbf{B}) = \mathbf{B}* \tag{9}$$

by simple set theory. Invariance of $\mathbf{A}$ under $\mathbf{w}^N$ gives us $\mathbf{A} \subset \mathbf{B}*$.

In the following chain,

$$\begin{aligned} h^N(\mathbf{A}, \mathbf{B}*) &= h^N(\mathbf{w}^N(\mathbf{A}), \mathbf{w}^N(\mathbf{B})) && (10) \\ &\leq \mathbf{s}h^N(\mathbf{A}, \mathbf{B}) && (11) \\ & && (12) \end{aligned}$$

we have equality (10) by invariance of $\mathbf{A}$ and definition of $\mathbf{B}*$. From this, the recurrent version of Hutchinson's lemma (Theorem 4.2) implies (11), where

$$\mathbf{s} = \max_{i=1\ldots N} \mathrm{Lip} w_i. \tag{13}$$

□

When creating a bounding volume hierarchy, simple sets, rather than set N-tuples, are used. The initial bounding volume corresponds to the initial vertex of the RIFS digraph (as specified in Chapter 7). Then, in the heap, each bounding volume must also contain the index of the most recently applied map from the RIFS. This information is used to find a "valid" map from the RIFS (as required in step 2.4 in Algorithm 12.4). As such, set N-tuples are incorporated into the standard instancing paradigm.

Figure 12.5 shows the bounding volume hierarchy for Sierpinski's tetrahedron. Only the top bounding volume is specified, the rest are generated automatically from the IFS and are valid and efficient by the bounding volume theorem.

12.7.2 Analysis

For the RIFS model, the diameter of the bounding volumes at each level n of the hierarchy is given by the left-hand side, and bounded from above by the right-hand side, of

$$\mathrm{diam} w_{i_n} \circ w_{i_{n-1}} \circ \cdots \circ w_{i_1}(B_0) \leq \mathbf{s}^n \mathrm{diam} B_0. \tag{14}$$

Thus the hierarchy traversal depth n is lower-bounded by

$$\log_{\mathbf{s}} \frac{p(t)}{\mathrm{diam} B_0} \leq n, \tag{15}$$

which is a variation on results derived in [Reuter, 1987; Hepting, 1991] for rendering 2-D linear fractals.

The bounding volume hierarchy produced by the RIFS is more efficient when neighboring bounding volumes in the hierarchy do not intersect often. Ideally, the RIFS should have the open-set property with the initial bounding volume set to the closure of open set.

Some attractors require overlapping constructions to be modeled efficiently. Other constructions may have the open-set property but with a complex open set, such as the twin-dragon, which posesses the open-set property with the open set equal to its interior. In general, it is better to use the simplest, smallest initial bounding volume available.

The most efficient bounding volumes are the canonical primitives such as the unit ball centered at the origin, or a likewise defined cone, cylinder, box. For such initial bounding volumes, sometimes the attractor must be affinely deformed and translated to the origin to snuggly fit inside the canonical primitive. Such attractors may be thought of as canonical attractors, which may then be transformed back into their intended form by an instancing operation.

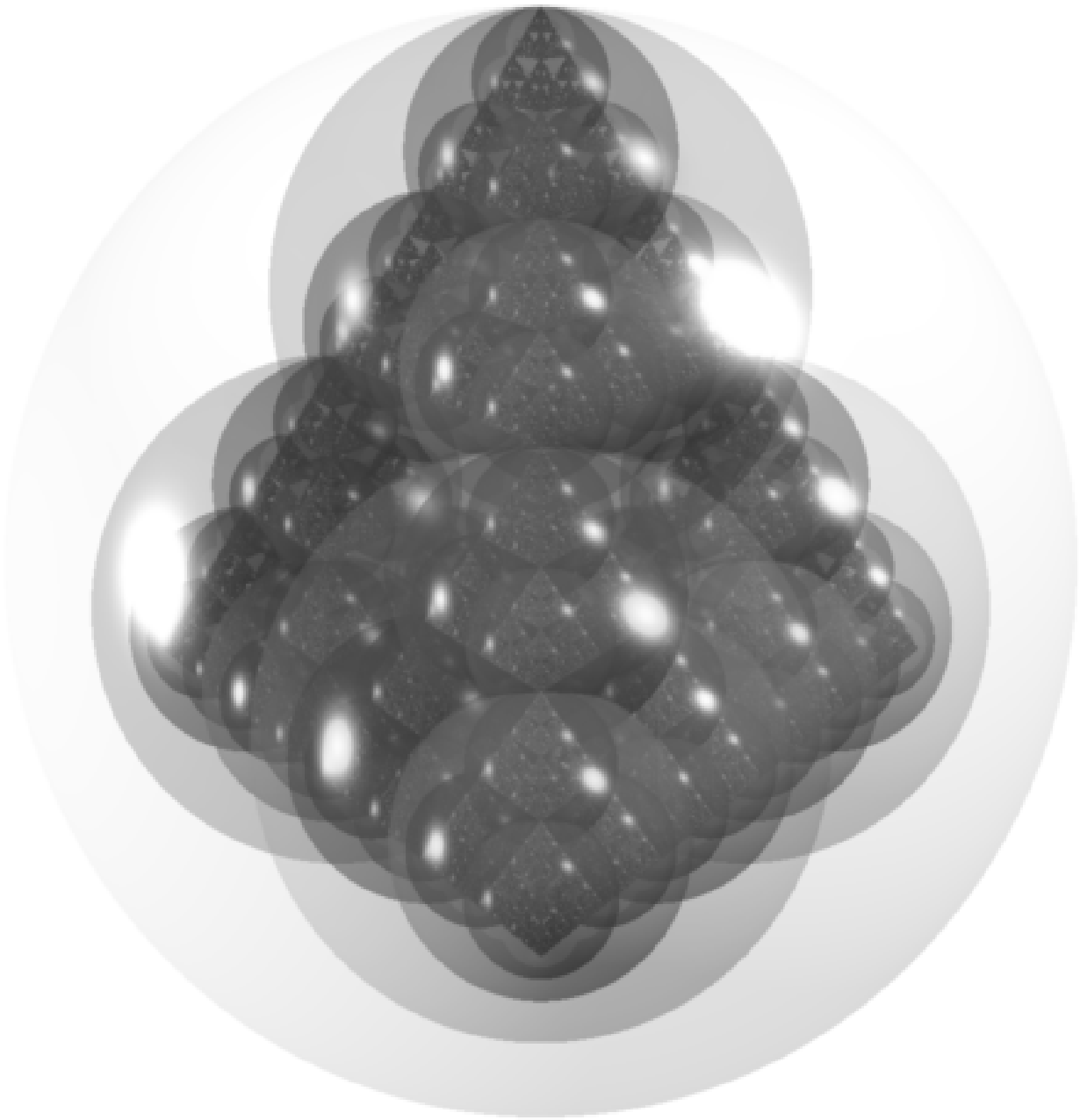

Figure 12.5: Hierarchy of bounding volumes for Sierpinski's tetrahedron.

Chapter 13

FAST RAY-LINEAR FRACTAL INTERSECTION

Suppose an RIFS $(\mathbf{w}, G)$ meets the following conditions:

(a) Every map $w_i \in \mathbf{w}$ is an affine map of the form

$$w_i(\mathbf{x}) = \mathbf{x} \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ a_1 & a_2 & a_3 & 1 \end{bmatrix} \tag{1}$$

and

(b) the RIFS satisfies the open-set condition.

This chapter shows that finding the first ray intersection with the 3-D attractor of such a RIFS is equivalent to deciding if a point is in the attractor of a 2-D version of the RIFS.

13.1 A Locally Orthogonal Approximation to Perspective

For each pixel in the image plane, let $R(\mathbf{o}, \mathbf{d})$ be a ray originating at the eyepoint extending through the center of the pixel. Mapping the results of each ray trace onto the resulting pixel comprises a perspective projection into the image plane.

Given each ray $R(\mathbf{o}, \mathbf{d})$, construct the following homogeneous 4×4 transformation matrix:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\mathbf{o}_1 & -\mathbf{o}_2 & -\mathbf{o}_3 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}_3 & 0 & \mathbf{d}_1 & 0 \\ 0 & 1 & 0 & 0 \\ -\mathbf{d}_1 & 0 & \mathbf{d}_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{\mathbf{d}_1^2 + \mathbf{d}_3^2} & \mathbf{d}_2 & 0 \\ 0 & -\mathbf{d}_2 & \sqrt{\mathbf{d}_1^2 + \mathbf{d}_3^2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

which transforms $R(\mathbf{o}, \mathbf{d})$ to the z-axis by translating the ray's origin to the space's origin, rotates the ray into the y-z plane, and finally rotates the ray into the z-axis [Foley *et al.*, 1990].

The orthogonal projection $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is defined by the transformation matrix

$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

which zeros the third coordinate of any vector its applied to. The application

$$\mathbf{x} M \Pi, \forall x \in R(\mathbf{o}, \vec{\mathbf{d}}) \tag{4}$$

maps the entire ray into the homogeneous 3-D origin:

$$R(\mathbf{o}, \mathbf{d}) \stackrel{M\Pi}{\longmapsto} (0, 0, 0, 1). \tag{5}$$

Thus the 3-D ray is projected to a single point, the origin, in 2-D. As far as the ray is concerned, this orthogonal projection is equivalent to a perspective projection.

13.2 Projected Recurrent Iterated Function Systems

Consider an RIFS $(\mathbf{w}, G)$ consisting of maps composed only of translations and uniform scales. Let $A \subset \mathbb{R}^3$ be its attractor and let $\pi_R(A) \in \mathbb{R}^2$ be its orthogonal projection onto a plane perpendicular to ray R.

Furthermore, let W_i be the homogeneous 4×4 transformation matrix equivalent of affine RIFS contraction w_i. Then one can construct a new 2-D RIFS consisting of maps denoted by the transformation matrices

$$\pi(W_i) = M^{-1} W_i M \Pi \tag{6}$$

which takes each point, maps it back to the original space, applies map w to it, maps it back to the orthogonal space, and then zeros out its third coordinate. Since W_i is of the form $s\mathbf{x} + \mathbf{a}$, where $s \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{R}^3$, losing the third coordinate does not affect the resulting projected attractor.

Let A_π be the attractor of this IFS. Then $\pi(A) = A_\pi$; the orthogonal projection of the attractor is the attractor of the orthogonal projection of the RIFS.

13.3 Ray Intersection via Point Inclusion

Let U be an open set satisfying the open set property of the 3-D RIFS and let U_π be its orthogonal projection. If the attractor is connected then $A \not\subset U$, though for all attractors $A \subset \overline{U}$. Therefore, let $B_\pi = \overline{U_\pi}$ be the bounding volume of A_π.

Furthermore, sort the RIFS maps such that $i \leq j$ implies that $d(\mathbf{o}, w_i(B)) \leq d(\mathbf{o}, w_j(B))$.

The ray $R(\mathbf{o}, \mathbf{d})$ projects to a point $\mathbf{r} \in \mathbb{R}^2$. Thus ray intersection problem is reduced to point inclusion. If $r \in A_\pi$ then ray $R(\mathbf{o}, \mathbf{d})$ intersects attractor A.

One can determine point inclusion by applying the maps of $\pi(\mathbf{w})$ to B_π. If one of the mappings excludes $\mathbf{r}$, then it is culled. Otherwise, the mappings are continually applied in

a depth-first manner, with priority on sorted order, until a mapping's contractivity factor is such that it reduces B_π to pixel scale, or point $\mathbf{r}$ is culled from all such composed mappings.

The ordering of the RIFS maps insures that bounding volumes occluding other bounding volumes will be searched first. The open-set condition insures that no point from a further bounding volume will occlude points from a nearer one. Hence, the first located hit will be the hit closest to the ray origin.

Chapter 14

INITIAL BOUNDING VOLUME CONSTRUCTION

Both ray-fractal intersection algorithms require an initial bounding volume B that contains the attractor of the RIFS. When one models an object as a linear fractal using the Collage Theorem, the bounding volume of the original object often suffices as the bounding volume of the linear fractal. When exploring the parameter space of linear fractals, one often has no *a priori* knowledge the shape or scale of the resulting attractor; one is required to bound an unknown object. In short, visualization of a linear fractal, in general, requires the algorithmic construction of an initial bounding volume.

The following discussions focus on the IFS model. Since the attractor of an RIFS $(\mathbf{w}, G)$ is embedded in the attractor of the IFS $\mathbf{w}$, the bounding volume of the attractor of the IFS also bounds the attractor of the RIFS, though perhaps not to the desired precision.

14.1 An Iterative Method

Let A be the attractor of an IFS $\mathbf{w}$. One can infer from Corollary 3.7 that if a bounding volume B has the property

$$B \supset \mathbf{w}(B) \tag{1}$$

then $B \supset A$.

Given any initial set B_0, repeated application of the Hutchinson operator will produce (by Corollary 3.7) the attractor. But the attractor is not a desirable bounding volume. Instead we have the following algorithm.

Algorithm 14.5 *Let* $\mathbf{w}$ *be an IFS with attractor* A. *Let* B_0 *be any ball* $B_{r_0}(x_0), r_0 > 0, x_0 \in \mathbb{R}^n$ *and let* $B_i = B_{r_i}(x_i)$ *be a sequence of balls of minimal radius* r_i *such that*

$$B_{r_i}(x_i) \supset \mathbf{w}(B_{r_{i-1}}(x_{i-1})) \tag{2}$$

with $x_i \in \mathbb{R}^n$ *dependent on* r_i. *In other words, each* B_i *is a ball whose diameter equals the diameter of the image of* B_{i-1} *under the Hutchinson operator. When centered properly,* B_i *contains* $\mathbf{w}(B_{i-1}$.

The sequence $\{B_i\}$ *converges to the limit*

$$B = \lim_{i \to \infty} B_{r_i}(x_i), \tag{3}$$

and, furthermore, $A \subset B$.

It seems one should be able to state this algorithm as a theorem and prove it. Unfortunately, the tools required for proof are very geometric in nature and beyond the scope of this dissertation[1].

[1] and the abilities of its author.

14.2 Computational Geometric Methods

The problem of finding a minimum bounding disk for a 2-D set is called the "Minimum Enclosing Circle Problem" [Preparata & Shamos, 1985]. Computation geometry concentrates on finite sets; the amount of time their solution takes to compute a set's smallest enclosing circle is a strictly increasing function of the number of points in the set. Unfortunately, fractals have uncountablely many points.

This hurdle is overcome by approximating the attractor with a finite set. One such set consists of the fixed points of finitely-many finite-length map compositions [Dubuc & Elqortobi, 1990]. First, let A_ϵ be the set of fixed points of finite map compositions (of minimal length) that take A to a set of diameter no larger than ϵ.

Definition 14.1 *Let* $\mathbf{w} = \{w_i\}_{i=1}^N$ *be an IFS. A point* $x \in A_\epsilon$ *if and only if there exists a sequence* $i_1, i_2, \ldots, i_k$ *such that*

$$x = w_{i_1,i_2,\ldots,i_k}(x) \tag{4}$$

and

$$\mathrm{diam}(w_{i_1,i_2,\ldots,i_{k-1},i_k}(A)) \leq \epsilon < \mathrm{diam}(w_{i_1,i_2,\ldots,i_{k-1}}(A)) \tag{5}$$

Second, we restate the following theorem from [Dubuc & Elqortobi, 1990].

Theorem 14.1 *Let* A *be the attractor of IFS* $\mathbf{w} = \{w_i\}$. *Then*

$$h(A, A_\epsilon) \leq \epsilon. \tag{6}$$

Thus, for the purpose of finding an optimal bounding ball, we approximate the uncountable set A with the finite set A_ϵ.

Now, given a finite set, any number of methods may be used to find a bounding ball. Two algorithms ($O(N \log N)$ and optimal $O(N)$) are given in [Preparata & Shamos, 1985] for the 2-D case. Both rely on Voronoi diagrams, whose complexity increases exponentially

with dimension. Rather than develop the sophisticated tools of computational geometry in 3-D, we instead choose an approximation which suffices for this purpose.

The approximation, from [Ritter, 1990], runs in $O(N)$ time and claims to approximate the minimum bounding ball within 5%.

Algorithm 14.6 *Let A_ϵ be the finite set approximation of attractor A, consisting of N points in $\mathbb{R}^n$. Then an approximate minimum bounding ball $B_r(o)$ can be found by the following steps.*

1. For each coordinate $i = 1 \ldots n$ define the two *points* $x^+_{(i)}$ and $x^-_{(i)}$ as $x^+_{(i)} = \max_{x \in A_\epsilon} x_i$ and $x^-_{(i)} = \min_{x \in A_\epsilon} x_i$.
2. Let (x^+, x^-) be the pair $(x^+_{(i)}, x^-_{(i)})$ of largest distance $|x^+_{(i)} - x^-_{(i)}|$.
3. Let $o = \frac{x^+ + x^-}{2}$ and $r = \frac{|x^+ - x^-|}{2}$.
4. For $x \in A_\epsilon \ldots$
4.1. If $|x - o| > r$ then ...
4.1.1. Let $\rho = \frac{r + |x-o|}{2}$.
4.1.2. Let $o = o + \frac{\rho(x-o)}{r|x-o|}$.
4.1.3. Let $r = \rho$.
4.1. End "If."
4. End "For."

Chapter 15

THE SIZE OF A PIXEL

The ray-fractal intersection algorithm needs to know the size of a pixel to determine if a bounding volume is to be treated as a primitive or not. Perspective distortion dictates that the diameter of an object's projection is proportional to its distance from the viewpoint. Hence, the size of a pixel is a linear function of the distance from the viewpoint to the ray-object intersection.

The idea of measuring the size of a pixel has appeared in numerous papers, though it wasn't until [Barr, 1986] that it was derived rigorously. We continue in that tradition with original results: bounding the [Barr, 1986] approximation and deriving similar results for shadows and planar reflections.

15.1 Eye-Ray Formulation

The horizontal extent of a pixel projected a distance t from the ray origin was originally approximated in [Barr, 1986] as

$$p(t) \approx \frac{2 \sin \frac{\theta}{2}}{N_h} \, t, \tag{1}$$

where θ is the field-of-view and N_h is the horizontal resolution. The frame buffer is assumed one unit from the ray origin.

Eq. (1) is actually the maximum horizontal extent of any pixel from the middle scan-line. The horizontal size of a pixel from any scan line is bounded by

$$\frac{2\tan\frac{\theta}{2}}{N_h\left(\frac{1}{\cos^2\frac{\theta}{2}}+\frac{\tan^2\frac{\theta}{2}}{A^2}\right)^{\frac{1}{2}}}t \le p(t) \le \frac{2\tan\frac{\theta}{2}}{N_h}\,t \tag{2}$$

(illustrated in Fig. 15.1), where A is the aspect ratio ($A = \frac{N_h}{N_v}$ if the pixels are square.)

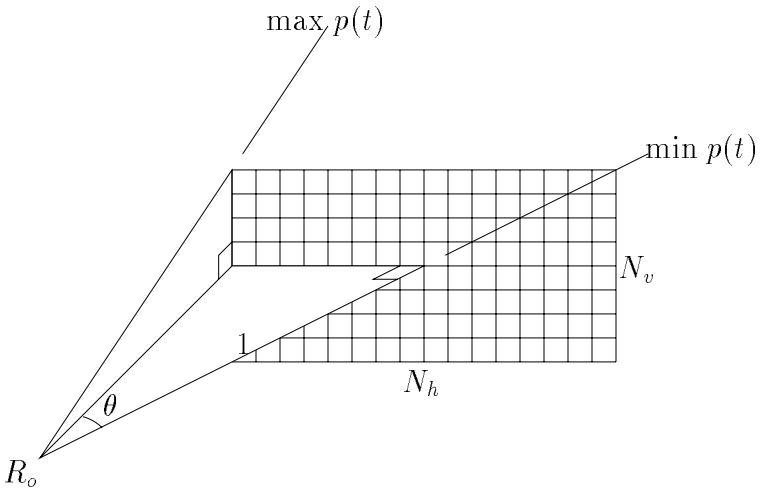

Figure 15.1: Pixel size geometry.

The exact size of a pixel at each pixel coordinate can be determined but, except for large viewing screens or head mounted displays where the field-of-view is large, it is usually much better to keep the size of a pixel constant for fixed t.

15.2 Light-Ray Formulation

If shadow rays are cast from the light source to the surfaces, then the size of a pixel can be used as a "closeness criterion" to determine if light rays reach the intersection point on the surface [Barr, 1986]. If the shadow rays are cast from the surface to the light source, then the ray's origin R_o can be translated by p in the ray's direction R_d to avoid immediate self-intersection.

The eye-ray formulation of pixel size does not apply to light rays. Two cases, where the resolutions of a fractal's shadow and the shadowed surface differ, illustrate this point.

(1) A distant light source illuminates a linear fractal, shadowing a closely inspected surface. Light rays cast from the light source to the surface will produce shadows of lower resolution than the surface while light rays cast from the surface to the light source will produce shadows of higher resolution than the surface.

(2) A light source illuminates a nearby linear fractal, shadowing a distant surface inspected from a viewpoint farther than the light—fractal distance but closer than the fractal—surface distance. Rays cast from the light source to the surface will produce a shadow of higher resolution than the surface. Conversely, rays cast from the surface to the light source create a shadow of lower resolution than the surface.

The light-ray pixel size is derived by setting the size of a pixel to zero at the light source and to the eye-ray pixel size at the surface. Let t_e be the distance from the eye to the surface and t_l be the distance from the surface to the light. If the rays are cast from the light source to the surface then the size of a pixel $p_l(t)$ at distance t from the light source, as shown in Fig. 15.2, is given by

$$p_l(t) = \frac{p(t_e)}{t_l}\ t. \tag{3}$$

If the rays are cast from the surface to the light source, then the size of a pixel at distance t from the surface is

$$p_l(t) = \frac{p(t_e)}{t_l}\ (t_l - t). \tag{4}$$

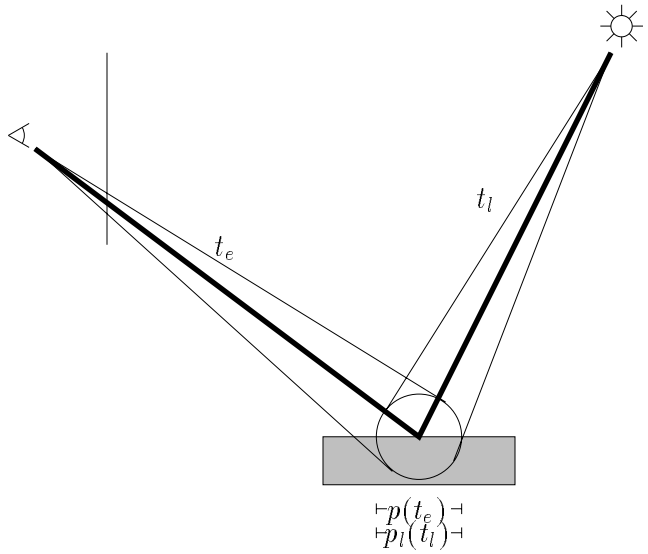

Figure 15.2: Light ray pixel size geometry.

15.3 Reflection/Refraction-Ray Formulations

Planar reflection causes a new ray to be generated. Thus, the formula for the size of a pixel from this new ray origin almost always differs from the previous formula.

The reflected size of a pixel is computed using a new viewpoint E_1 found by reflecting the original viewpoint, denoted as E_0, across the plane $P = (a, b, c, d)^T$. Let $N = (a, b, c)^T$ be the unit normal vector of plane P and let $\mathbf{x} = R_o + tR_d$ be the point that ray R intersects plane P. The new viewpoint is then

$$E_1 = E_0 - 2|N \cdot (E_0 - \mathbf{x})|N. \tag{5}$$

Then the reflected size of a pixel is given by

$$p_r(t) = \frac{p(t_e)}{|R_o - E_1|}\ (t + |R_o - E_1|), \tag{6}$$

where R_o is the origin of the reflection ray and $p(t_e)$ is the size of a pixel at the previous ray-plane intersection point R_o.

Reflection and refraction from curved surfaces can change the size of a pixel dramatically. A ray tracing microscope, constructed out of refractive solids, is alluded to in [Barr, 1986]. Even though this magnifying glass would increase the sampling resolution of the magnified object, the size of a pixel would not be dramatically decreased and the sub-pixel primitives would be magnified and become visible.

Chapter 16

THE LIPSCHITZ CONSTANT OF AFFINE MAPS

16.1 Estimating the Lipschitz Constant

Since the diameter of a bounding volume is tested against the diameter of a pixel, in most cases, an estimate of the bounding volume's diameter suffices. These estimates are much easier and faster to compute than the actual value for the Lipschitz constant of an affine map.

16.1.1 A Lower Bound

The lower bound of the Lipschitz constant of the composition of affine maps (whose Lipschitz constants are know *a priori*)[1] is the product of the Lipschitz constants of the individual maps

$$\mathrm{Lip} w_1 \circ w_2 \cdots w_k \leq \prod_{i=1}^{k} \mathrm{Lip} w_i. \tag{1}$$

[1]Often affine transformations are constructed from parameterized canonical affine transformations whose Lipschitz constants are trivially derived.

If the w_i are similtudes, then the inequality in (1) can be replaced with equality. For general affine maps w_i, this method provides only an upper bound, one which can be miserable in some extreme cases, as shown explicitly in [Hepting *et al.*, 1990].

16.1.2 An Upper Bound

Let W be the 4×4 homogeneous transformation matrix representation of an affine map $w : \mathbb{R}^3 \to \mathbb{R}^3$. Then its determinant is the change in volume of the set

$$\det W = \frac{\mathrm{vol}(w(B))}{\mathrm{vol}(B)} \tag{2}$$

where $B \subset \mathbb{R}^3$. Since volume of a set is proportional to the cube of its diameter, with this proportion maximal when the set is a ball, we have the lower bound

$$\mathrm{Lip} w \geq \sqrt[3]{\det W}. \tag{3}$$

As before, if the w_i are similtudes, then the inequality in (3) can be replaced with equality. In general, this method provides only a lower bound.

16.2 Computing the Lipschitz Constant

The most accurate method for determining the Lipschitz constant of a homogeneous affine transformation matrix M is by using a polar decomposition.

16.2.1 Polar Decomposition

Let T be the linear part (the upper-left 3×3 submatrix) of M. Then T, being real, square and invertible, can be decomposed into two component parts

$$T = QS \tag{4}$$

where Q is orthogonal and S is a symmetric positive definite matrix [Strang, 1988]. Component Q orients, S scales. We are concerned with S.

The component S is found from the derivation

$$\begin{aligned} T^T T &= (QS)^T QS && (5)\\ &= S^T Q^T QS && (6)\\ &= SQ^{-1}QS && (7)\\ &= S^2. && (8) \end{aligned}$$

The Lipschitz constant of w is found as the largest eigenvalue of S.

16.2.2 Computing Eigenvalues

The matrix $T^T T$ is symmetric and so is S^2. The eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of S^2 are real and can be found algorithmically using Jacobi transformations [Press *et al.*, 1988]. The resulting Lipschitz constant of w is thus found as

$$\mathrm{Lip} w = \max_{i=1,2,3} \sqrt{\lambda_i}. \quad (9)$$

Chapter 17

HIERARCHICAL SHADING

As mentioned in Chapter 11, surface normals are undefined on fractals. In order to visually perceive a 3-D object, one depends heavily on shading cues to represent surface orientation. This chapter proposes a shading approximation for fractal surfaces based on their hierarchical representations.

17.1 Formulation

Consider a natural linear fractal: cauliflower. The surface of a cauliflower is made of an extremely large number of small buds. Shading a surface containing such tiny buds will likely exceed the Nyquist limit and will be prone to aliasing [Shannon, 1949].

Hierarchical shading is based on the idea that the cauliflower surface reflects light diffusely as a sphere since the small buds loosely approximate the surface of a sphere. The cauliflower reflects light more like several medium-sized spheres since the buds more closely approximate them. The illumination of many smaller spheres even more accurately represents the light reflected by these buds.

17.2 Diffuse Illumination

Until the reflectance properties of fractal surfaces are better understood, they should be rendered diffusely. Specular reflections produce highlights, which are a useful cue to the perception of surface orientation. Unfortunately, highlights also give the perception of smooth surfaces, which fractals necessarily do not have.

Using Lambert's law, the illumination color $\mathbf{c}(\mathbf{x})$ of a surface point $\mathbf{x}$ is computed as

$$\mathbf{c}(\mathbf{x}) = k_a \mathbf{c}_a \mathbf{c}_x + k_d \sum_{i=1}^{L} (\vec{\mathbf{n}} \cdot \vec{\mathbf{l}}_i) \mathbf{c}_i \mathbf{c}_x \tag{1}$$

where k_a and k_d are the respective percentages of ambient and diffuse contributions, $\mathbf{c}_a, \mathbf{c}_x$ and $\mathbf{c}_i$ are the respective colors of the ambient light, the surface and the light sources, $\vec{\mathbf{n}}$ is the surface normal at point x and $\vec{\mathbf{l}}_i$ is a unit vector in the direction of the ith light source from point $\mathbf{x}$.

The light vector $\vec{\mathbf{l}}_i$ is uniform across the surface for a "directional" light source. The light vector varies across the surface for a "point" light source and is constructed for each point $\mathbf{x}$ on the surface as

$$\vec{\mathbf{l}}_i = \frac{\mathbf{o_i} - \mathbf{x}}{|\mathbf{o_i} - \mathbf{x}|} \tag{2}$$

where o_i is the location of the point light source.

17.3 Definition

Hierarchical shading is created as the weighted sum of the shading of the bounding volumes containing a surface. Suppose ray $R(\mathbf{o}, \vec{\mathbf{d}})$ intersects the surface at point $\mathbf{x}$. Let B_i be a descending sequence of bounding volumes that converge to point $\mathbf{x}$. Then the hierarchical shading of surface point $\mathbf{x}$ is computed as the weighted sum

$$\mathbf{c}_h(x) = \frac{\sum_{i=1}^{\infty} w(\mathrm{diam}(B_i)) \mathbf{c}(\mathbf{y}_i)}{\sum_{i=1}^{\infty} w(\mathrm{diam}(B_i))} \tag{3}$$

where $w : \mathbb{R} \to \mathbb{R}$ is a weighting function, $\mathbf{y}_i$ is the point ray $R(\mathbf{o}, \vec{\mathbf{d}})$ intersects bounding volume B_i and $\mathbf{c}(\mathbf{y}_i)$ is the illumination at point $\mathbf{y}_i$ as defined by (1).

One shortcut in the computation of (3) is to compute a "total" normal as the weighted sum the surface normals. Hierarchical shading may then be approximated by diffusely illuminating point x by this total normal. This approximation is accurate for directional light sources (where the light vector is uniform across the surface) but errors proportionate to the diameter of the bounding volumes for point light sources (where the light vector varies across the surface).

17.4 Weighting Methods

The choice of weighting function directly affects the appearance of the hierarchical shading. Three linear weighting functions are offered here, though other functions, perhaps non-linear, may be required to provide a desired effect.

17.4.1 Constant Weighting

The constant weighting function sums the shading of all encountered bounding volumes uniformly so that the shading of the initial bounding volume contributes as much as the shading from a terminal bounding volume. The constant weighting function is defined simply as

$$w(x) = 1. \tag{4}$$

A graph of this weighting function for an RIFS with contractivity factor $\mathbf{s}$ is shown in Figure 17.1a.

Though the shading of all bounding volumes is weighted uniformly, the constant weighting is not uniform across scale; the shading is biased toward the smaller bounding volumes.

17.4.2 Low-Pass Weighting

The low-pass weighting function removes the scale bias of constant weighting by weighting the shading of smaller bounding volumes less than the shading of larger bounding volumes. It is defined as

$$w(x) = x \tag{5}$$

and its graph is shown in Figure 17.1b.

17.4.3 High-Pass Weighting

The high-pass weighting function dampens the surface normals of larger bounding volumes and emphasizes the surface normals of the smaller bounding volumes. It is defined as

$$w(x) = \text{diam}(B_0) - x \tag{6}$$

where B_0 is the initial bounding volume. Its graph is illustrated in Figure 17.1c.

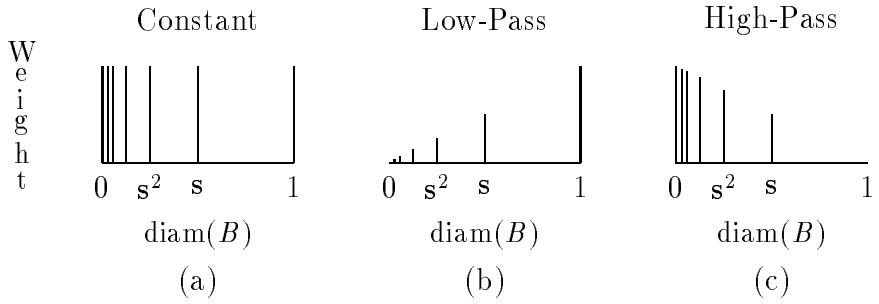

Figure 17.1: Shading weight functions for (4), (5) and (6), respectively.

17.5 Analysis

Hierarchical shading can be compared to standard Lambertian shading by modeling a Euclidean surface with an RIFS. One such model is an extruded Sierpinski's gasket. This hybrid shape looks like the standard planar Sierpinski's gasket when viewed down the axis of extrusion, but when viewed perpendicular to this axis, it reveals a planar surface.

Three extruded Sierpinski's gaskets appear in Figure 17.2 corresponding to the weights from (4), (5) and (6), respectively.

The high-pass weighting produces moiré patterns which are almost completely suppressed by the constant weighting. The low-pass weighting does not reveal much detail, particularly from the fractal side of the middle extruded Sierpinski's gasket.

The analysis consists of comparing the hierarchical shading from each of the three types of weighting functions on the planar surface of the extruded Sierpinski's gasket with the normal of a similarly oriented plane. Let $\vec{\mathbf{n}}_c$ be the surface normal approximated by the high-pass weighting function (4), let $\vec{\mathbf{n}}_l$ be the surface normal approximated by the low-pass weighting function (5), and $\vec{\mathbf{n}}_h$ be the surface normal approximated by the high-pass weighting function (6). Furthermore, let $\vec{\mathbf{n}}_p$ be the normal of a similarly oriented plane.

The discrepancies between the hierarchically defined surface normals and the analytically defined surface normals for $10,000$ samples were measured by their dot product and plotted in Figure 17.3.

The constant and high-pass weighting functions produced very narrow distributions about the central value whereas the distribution of the low-pass weighting function is broader. Each distribution had mode one; the most common hierarchically approximated surface normal for this surface is the analytically defined surface normal. The standard deviations of the three distributions about this one central value were .036, .084 and .031, respectively.

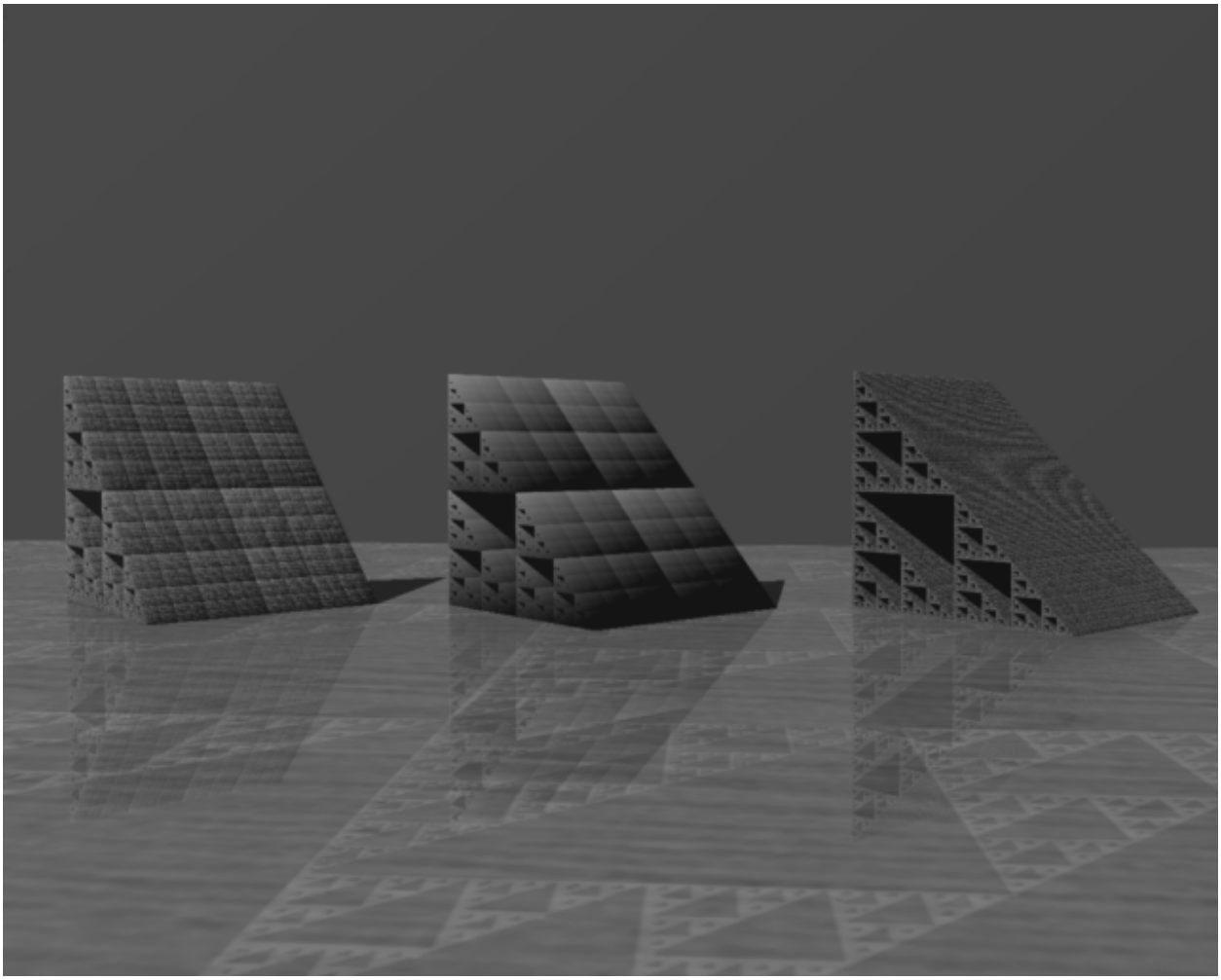

Figure 17.2: Extruded Sierpinski's gaskets shaded by constant (left), low-pass (middle), and high-pass (right) shading modules.

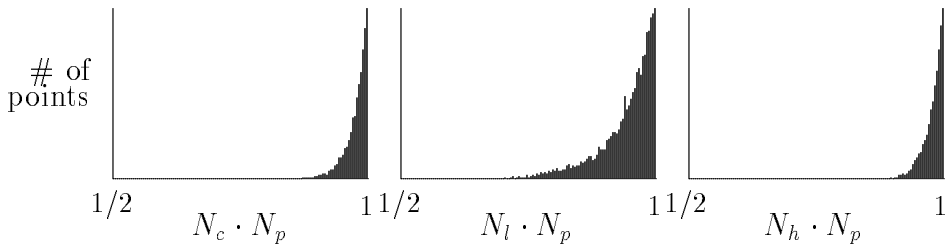

Figure 17.3: Distribution of surface normals for Eqs. (4), (5) and (6).

17.6 Evaluation

Since the bounding volumes a ray intersects vary depending on the direction the ray is coming from, the hierarchical shading of the same surface may appear different when viewed from different angles. In the experience of creating several animations, this artifact has never blatantly presented itself, though subtle changes are sometimes noticed.

Hierarchical shading is by no means an analytically correct method for shading fractal surfaces. The actual surface normal of a fractal is likely multivalued: a distribution of surface normals that must be integrated for proper shading computations. Nonetheless, hierarchical shading does a good job of cueing the orientations of fractal surfaces. It proper use is for visualization and was not intended to produce analytically correct renderings.

Chapter 18

LOCAL COVERS

Aliasing is a current problem in computer graphics for which many solutions have been offered. It is no less a problem in the rendering of fractal sets. Indeed, the detail these sets offer is much more than most antialiasing algorithms expect.

This chapter extends the area sampling methods of [Thomas *et al.*, 1989] into the author's original work regarding the rendering of linear fractals, though the results apply to any number of other fractal and non-fractal models.

18.1 Covers

A cover is a set of two bounding volumes that are a pixel's width larger and smaller than the set [Thomas *et al.*, 1989].

Definition 18.1 *A* cover *of a set* $A \subset \mathbb{R}^n$ *and ray* $R(\mathbf{o}, \vec{\mathbf{d}})$ *is the pair* $(B^+, B^-) \in (\mathbb{R}^n)^2$ *defined*

$$B^+ = A + p(t_0) \tag{1}$$

$$B^- = A \setminus (\partial A + p(t_0)) \tag{2}$$

where $t_0 = t_0(R(\mathbf{o}, \vec{\mathbf{d}}), A)$.

Rays are tested against the outer cover B^+, then the set A and then the inner cover B^-. If the ray intersects the outer cover but not the inner cover then a sillhouette edge passes through the pixel in the image plane. When this happens, the pixel should receive some portion of illumination from the surface, the remainder coming from the further exploits of the ray.

In [Thomas *et al.*, 1989], the minimum distance from the ray to the surface

$$d = \inf\{|x - y| : x \in R, y \in B\} \tag{3}$$

is used to interpolate the illumination as

$$k(d) = \begin{cases} 0 & \text{if } d > p(t_0) \text{ and } R(\mathbf{o}, \vec{\mathbf{d}}) \text{ misses } A, \\ 0 & \text{if } d > p(t_0) \text{ and } R(\mathbf{o}, \vec{\mathbf{d}}) \text{ misses } A, \\ \frac{p(t)-d}{2p(t_0)} & \text{if } d \leq p(t) \text{ and } R(\mathbf{o}, \vec{\mathbf{d}}) \text{ misses } A, \\ \frac{p(t)+d}{2p(t_0)} & \text{if } d \leq p(t) \text{ and } R(\mathbf{o}, \vec{\mathbf{d}}) \text{ hits } A, \\ 1 & \text{if } d > p(t_0) \text{ and } R(\mathbf{o}, \vec{\mathbf{d}}) \text{ hits } A \end{cases} \tag{4}$$

which is a piecewise linear ramp function.

Shadow sillhouettes on the surface are similarly blended, creating smoother, slightly softer shadows on surfaces. However, these shadows may still be aliased on the image plane depending on the orientation of the shadow surface. Here, [Thomas *et al.*, 1989] suggests projecting the distance between the shadow ray and the shadowing surface onto the shadowed surface (approximated by a tangent plane). Then project this distance onto the image plane to find the proper blending proportion.

Reflected and refracted sillhouettes are treats in the same way, though simulating the reflecting/refracting surface with a tangent plane is not accurate for surfaces of high curvature. In this case, the authors of [Thomas *et al.*, 1989] suggest other forms of antialiasing.

A transfer function is used to blend the color of a surface with its background. Several transfer functions are available from the area of volume rendering [Kajiya, 1983], etc.; the

simplest, used here, are the linear approximations used for image compositing [Porter & Duff, 1984].

If $\mathbf{c}$ and α are the current encountered ray color and opacity and $\mathbf{c}_A$ and α_A are the color and opacity of shape A, then the linear transfer functions are, for source to surface ray traversal,

$$\mathbf{c} = \mathbf{c} + (1-\alpha)\alpha_A \mathbf{c}_A, \tag{5}$$

$$\alpha = \alpha + (1-\alpha)\alpha_A, \tag{6}$$

and, for surface to source ray traversal (sometimes used for light rays),

$$\mathbf{c} = (1-\alpha_A)\mathbf{c} + \alpha_A \mathbf{c}_A m \tag{7}$$

$$\alpha = \alpha + (1-\alpha)\alpha_A. \tag{8}$$

One main drawback of covers has not received a fair treatment in the literature (including [Hart & DeFanti, 1991]): each intersecting cover requires rays to be cast to each light source for illumination. This can be quite tedious for rays that graze many surfaces. Fortunately, this happens infrequently.

The most major drawback of covers is their limitation to simple geometries. Section 7 of [Thomas *et al.*, 1989] constructs covers for spheres, ellipsoids, planes, splines, CSG objects and polyhedra. Nonetheless, highly detailed geometries, created from such techniques as micropolygons or displacement maps, are very difficult to cover efficiently. Fractal surfaces are exceptionally problematic for such object methods. The next section solves this problem by covering a highly detailed object by concentrating on the sections near the ray.

18.2 Local Covers

A local cover covers a single primitive; their union over the object forms a cover as described above. If the primitives are organized hierarchically using bounding volumes, then the a

bounding volume from the hierarchical ancestry of a primitive can be used as its local cover.

Definition 18.2 *A bounding volume B in an hierarchy of bounding volumes $\mathcal{B}$ is a* local cover *of a point on a surface if and only if*

(a) B intersects R,

(b) The diameter of B falls within preset bounds τ_{min} and τ_{max}, and

(c) none of B's children in the hierarchy intersect R.

The values $\tau_{\min}$ and $\tau_{\max}$ denote the bandwidth of the local covers. They are most effective when set proportional to the size of a pixel $p(t)$. The bandwidth is used to avoid computing the minimum distance between the ray and the surface.

The upper limit of the bandwidth, $\tau_{\max}$, is the scale at which bounding volumes take on non-zero opacity. It resembles the value r from [Thomas *et al.*, 1989]. We suggest setting it to $2p(t)$, twice the size of a pixel.

The lower limit, $\tau_{\min}$, is the scale at which bounding volumes are completely opaque; no further subdivision is performed. Intersecting such a bounding volume is reminiscent of intersecting the inner cover from [Thomas *et al.*, 1989] at a distance greater than r from the surface. Section 5 of [Barr, 1986] suggests setting $\tau_{\min}$ to $\frac{1}{20}p(t)$, which appears good.

As before, a simple piecewise linear function of diameter produces the desired opacity values for local covers,

$$\alpha_B = \begin{cases} 1 & \text{if } \mathrm{diam}(B) < \tau_{\min}, \\ 1 - \frac{\mathrm{diam}(B) - \tau_{\min}}{\tau_{\max} - \tau_{\min}} & \text{if } \tau_{\min} \leq \mathrm{diam}(B) < \tau_{\max}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

18.3 Evaluation

An example of the results of antialiased rasterization using local covers can be seen in Figure 18.1. This figure is an enlargement of a 32 by 32 pixel area from Figure 20.1. Notice the blending into the background of the hard diagonal edge.

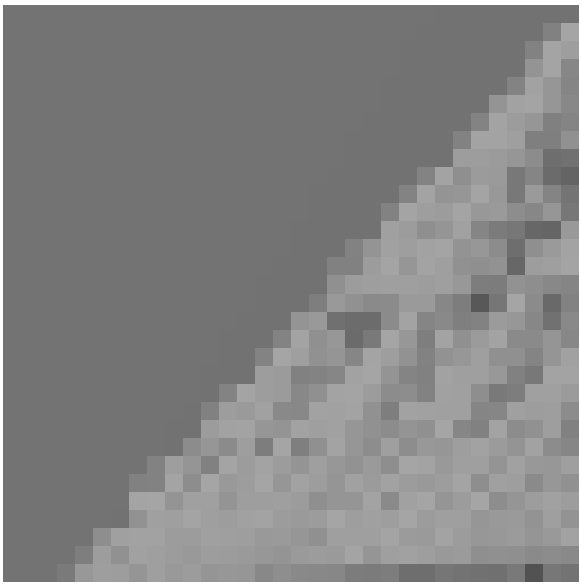

Figure 18.1: Detail of antialiased rasterization using local covers.

The animation [Hart, 1991] was plagued with aliases in its depiction of grass, due to several reasons. The first was that there was not enough memory available in the Pixel Machine to find the correct Lipschitz constant computation. Instead, the determinant was used and the grass transformations, which squashed and stretched the entire field of grass into each blade, were far from similtudes.

The second reason was that the field-of-grass-to-a-single-blade transformation had a small Lipschitz constant (with even smaller determinant). This meant that few, if any, bounding

volumes fell within the local cover bandwidth. Thus, proper blending proportions were rarely found, resulting in an all-or-nothing rasterization.

Finally, the special technique for shadow silhouettes developed in [Thomas *et al.*, 1989] used a tangent plane surface approximation, and so is not applicable to fractal surfaces. Hence, extremely rough surfaces (with Hausdorff dimension approaching 3), such as grass, will likely produced heavily aliased shadow silhouettes.

Covers are an approximation. Local covers approximate covers. Both are based on the assumption that the proportion of the pixel filled with a shape's projection is related to the distance from the ray to the surface. It is easy to construct convoluted counter-examples where this assumption is not true. Furthermore, fractal surfaces often test approximation techniques harshly.

Local covers, like [Thomas *et al.*, 1989] covers, do not produce area samples as accurately as supersampling techniques. They are, however, quite efficient, requiring only one eye ray per pixel.

18.4 Extensions and Applications

The bounding volume hierarchy itself may be visualized by setting $\tau_{\max} = \infty$ and using a constant fractional opacity value. This makes all of the bounding volumes uniformly translucent. An example of this appears in Fig. 12.5.

The algorithms in [Kajiya, 1983; Bouville, 1985] are examples of stochastic instancing, where the instances of each bounding volume are randomly perturbed. The same kind of bounding volume hierarchy is used for these random fractals as is used for linear fractals. Hence, the smaller "cheesecake extents" and "bounding ellipsoids" may be used as local covers for antialiasing these fractal terrain models.

In [Nishita *et al.*, 1990] (a recent improvement to an algorithm in [Rubin & Whitted, 1980]), rational bezier surfaces are subdivided, creating a hierarchy of bounding volumes

extending down to the point level. Ironically, the same algorithm that antialiases fractal surfaces can antialias smooth surfaces as well.

In [Hart *et al.*, 1989], quaternion Julia sets were rendered using a distance estimate to compute the radius of so-called "unbounding volumes." When rays graze the surface, the diameter of these unbounding volumes reduces as the bounding volumes in the linear fractal case. Hence, the diameter of the unbounding volume can be used as the diameter of a bounding volume in the opacity computation. Local covers may be simulated for quaternion Julia sets.

When any complicated database is viewed from a distance, it is phenomenologically fractal — that is, interesting at all visible levels of detail. Thus, the bounding volume hierarchy may be pruned at a higher level than the primitive level to save computation time. This technique is given a thorough treatment in [Thompson, 1991].

Part V

CONCLUSION

Chapter 19

IMPLEMENTATION

These algorithms and methods were developed on an AT&T Pixel Machine 964dX [Potmesil & Hoffert, 1989]. The Pixel Machine is a parallel MIMD image computer consisting of 64 AT&T DSP32 digital signal processors. The frame buffer is distributed among the 64 processors such that every eighth pixel in the horizontal and vertical directions are connected to the same processor, but to no other processor. Serial communication is available along a toroidal topology between processors but was not used in this implementation.

The AT&T DSP32 digital signal processor is a single precision pipelined processor with a peak performance of 10 MFLOPS and 5 MIPS. The MFLOPS rating is twice the MIPS rating since a single instruction can perform a multiplication and an add. Often, the compilers for such processors do not take full advantage of their capabilities. Hence, much of the code for these processors was written at a low level, often with assembler directives in the "C" code.

The Pixel Machine's memory architecture, under the DEVtools operating system, contains 36kB of available programming space, limiting the size of programs significantly. Lower speed data space is also available, in three large chunks which are commonly used for the frame buffer, the texture space and the z-buffer.

In the implementation of the previously described algorithms, the 36kB programs space

constraint was overcome by the following. The heap, containing the transformation matrices, top level bounding volume pointers, and color and shading information, was kept in the z-buffer memory, with a maximum of 256 entries. The shape database, containing the iterated function systems and object databases were stored in the back frame buffer. The texture memory was left available for 2-D texture maps.

A program on the host, a Sun 4, interpreted modeling files and passe the derived raw data to the Pixel Machine for rendering. Upon completion, the rendered image could be saved in a file for later redisplay.

Chapter 20

EXHIBITION

Two examples, showing the range of linear fractals, from visualization to image synthesis, from the geometric to the natural, from inside to the outdoors, are described and exhibited. Color versions may be found in [Hart & DeFanti, 1991]. The iterated function system codes used to produce these images may be found in Appendix B.

20.1 The Five Non-Platonic Non-Solids

The five non-Platonic non-solids are a satire of the five Platonic solids: the tetrahedron, octahedron, hexahedron, icosahedron and dodecahedron. The five non-Platonic non-solids are: Sierpinski's tetrahedron, octahedron and icosahedron, Menger's sponge and von Koch's snowflake-a-hedron. These shapes are non-Platonic for their fractal, non-Euclidean, geometries. They are non-solids since each one has infinite surface area and/or zero mass.

Sierpinski's tetrahedron and octahedron, and Menger's sponge have the open-set property. Sierpinski's icosahedron and von Koch's snowflake-a-hedron do not have the open-set property but do exhibit this property on their faces. Sierpinski's gasket may be found in the woodgrain floor.

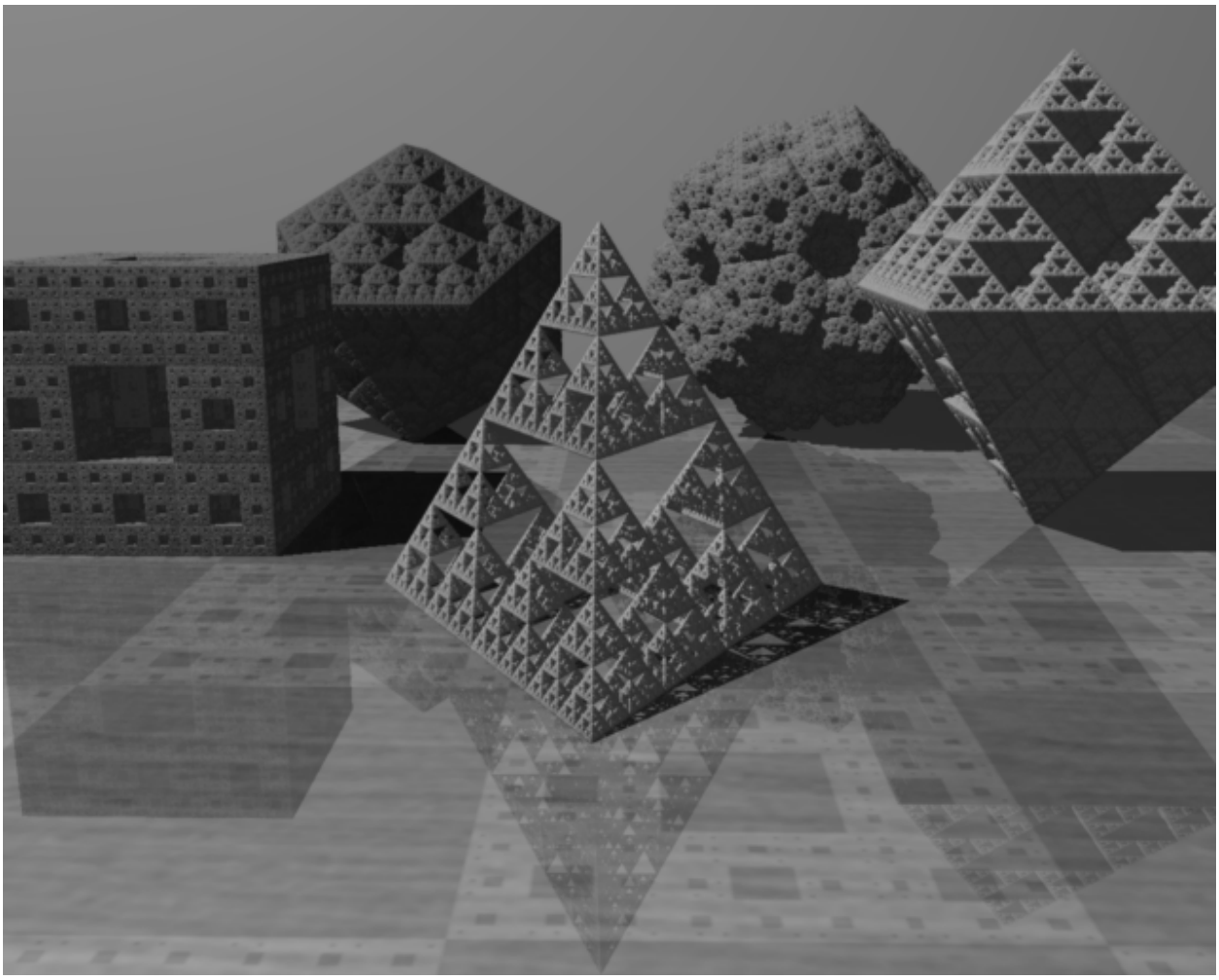

Figure 20.1: The five non-Platonic non-solids.

20.2 Fractal Forest

The fractal forest demonstrates the power of linear fractals in modeling nature. In this scene, every leaf, pine needle, piece of bark and blade of grass is accounted for in the linear fractal model and those visible are rendered.

The elm trees are composed of a trunk with four smaller elm trees extending from it. The trunk itself consists of three small stretched upright elm trees and three small stretched inverted elm trees. The bark is actually the stretched image of the foliage.

The pine trees consist of six base branches and one "rest of the pine tree" section. The pine tree is cone-shaped, as are its branches. Real pine branches are thin at the base and become larger at their ends. These pine trees were modeled with IFS models; an RIFS model would do a better job.

Finally, as mentioned at the end of Chapter 18, the field of grass is tiled using a twin-dragon shape. The blades themselves are actually themselves fields of grass, though compressed and stretched to form individual blades.

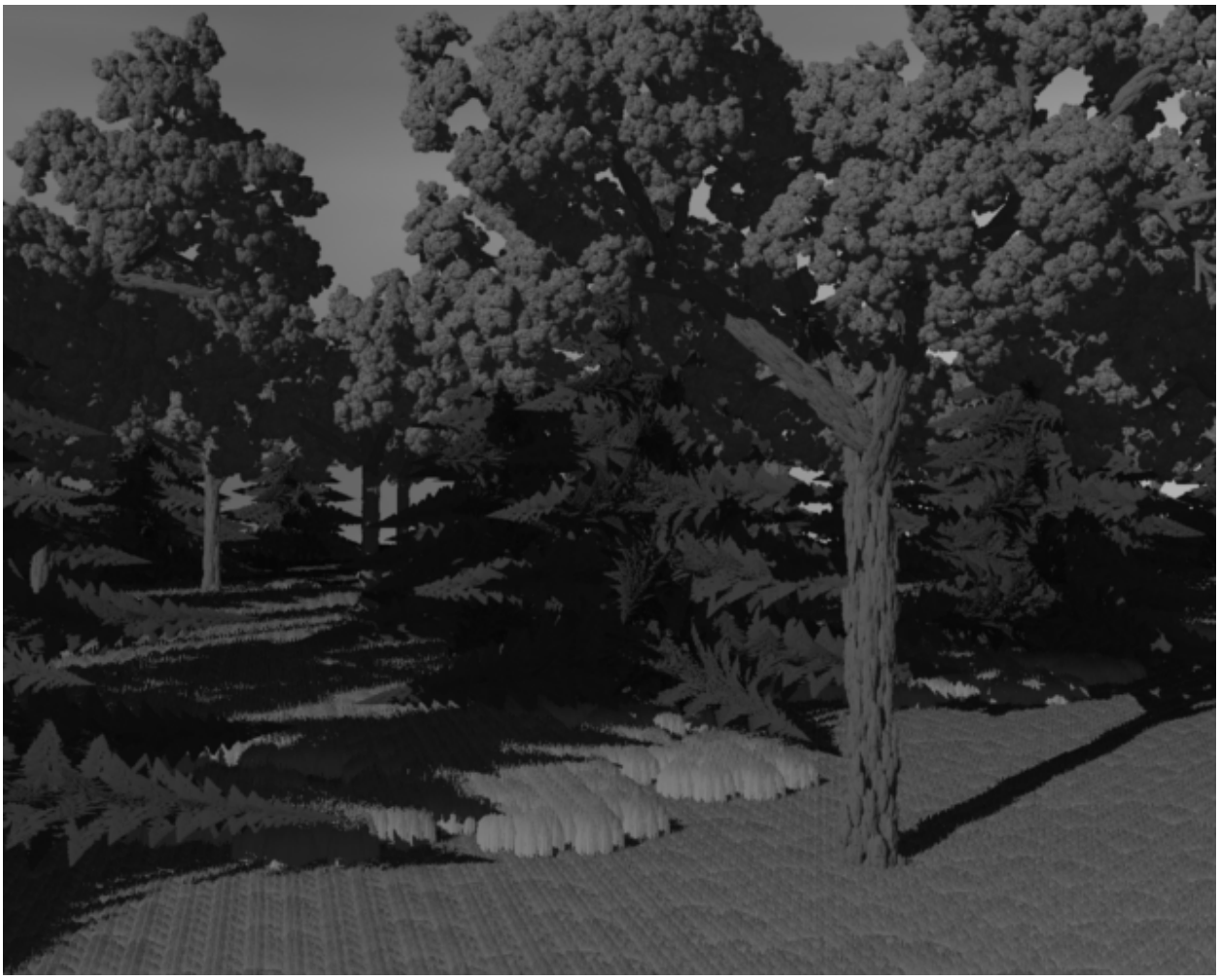

Figure 20.2: The fractal forest.

Chapter 21

FURTHER RESEARCH

This dissertation is by no means a complete solution to the problems of rendering fractals. The following areas are in much need of attention.

This dissertation concentrates on linear fractals. As detailed in the introduction, many other kinds of fractals exist. Only a handful of methods have been developed for rendering or otherwise visualizing fractal sets in 3-D. Other fractal sets whose research would benifit from efficient rendering methods are the many strange attractors from physics, the cubic connectedness locus [Milnor, 1991] and the family of deterministic fractal sets in the quaternions [Norton, 1989b], to name a few.

The section on modeling surveys methods for automatically modeling objects as linear fractals. This specific area has vast implications in the area of data compression and is still much in its infancy.

The problem of finding an optimal initial bounding volume remains largely unsolved, though does not appear to be significantly difficult.

The shading of fractal surfaces has yet to receive a thorough, rigorous treatment. Fractal surfaces lack tangent planes — surface normals — though it can be shown that a distribution of surface normals more accurately portrays the reflectance properties of fractal surfaces.

Furthermore, it is likely that extruded fractal surfaces, such as those found on quaternion Julia sets and some strange attractors, reflect light anisotropically. Until more research is done, Lambert's diffuse reflection model is recommended when rendering fractal surfaces.

Finally, the problem of aliasing is particularly problematic with fractal sets. The use of local covers efficiently inhibits aliases. Nonetheless, the self-similarity of fractal sets should be exploited to better compute the rasterization integral, eliminating aliases with a more accurate approximation.

APPENDICES

Appendix A

METRIC SPACE ESSENTIALS

This appendix derives the basics of metric spaces used in the dissertation, for the benefit of those readers not fortunate enough to have taken the course. It is based largely on [Kaplansky, 1977].

Definition A.1 *A metric space is an ordered pair* $(\mathbb{X}, d)$ *where* $\mathbb{X}$ *is a set (space) and* $d : \mathbb{X}^2 \to \mathbb{R}$ *is a real function (metric) satisfying*

$$\begin{aligned} d(x,x) &= 0, \forall x \in \mathbb{X}, & (1)\\ d(x,y) &> 0, \forall x, y \in \mathbb{X}, x \neq y, & (2)\\ d(x,y) &= d(y,x), \forall x, y \in \mathbb{X}, & (3)\\ d(x,z) &\leq d(x,y) + d(y,z), \forall x, y, z \in \mathbb{X}. & (4) \end{aligned}$$

The Euclidean metric on space $\mathbb{R}^n$ is defined as the root of the sum of squares, namely

$$d(x,y) = \left(\sum_{i=1}^{N} (x_i - y_i)^2 \right)^{\frac{1}{2}}. \quad (5)$$

We will almost always use the Euclidean metric.

The only other metric used here is the chessboard metric, defined

$$d(x,y) = \max_{i=1\ldots N} x_i - y_i. \quad (6)$$

The difference between the two metrics can be observed by drawing curves of equal distance about a point. In $\mathbb{R}^2$ the Euclidean metric yields a circle, the chessboard metric, a square. Such a curve and the regions it bounds is called a "ball."

Definition A.2 *The ball $B_r(x)$ of radius r about a point x in metric space $(\mathbb{X}, d)$ is defined*

$$B_r(x) = \{y : d(x, y) \leq r\}. \tag{7}$$

Let $U_r(x)$ be defined similarly to (7) but with the strict inequality

$$U_r(x) = \{y : d(x, y) < r\}. \tag{8}$$

Then U_r is an "open" set and is used to define all open sets of a metric space.

Definition A.3 *A set U of metric space $(\mathbb{X}, d)$ is* open *if and only if for each point $x \in U$ there exists a positive r such that*

$$U_r(x) \subset U. \tag{9}$$

Theorem A.1 *Let $\mathcal{U}$ be any collection of open sets of metric space $(\mathbb{X}, d)$. Then*

$$V = \bigcup_{U \in \mathcal{U}} U \tag{10}$$

is open.

Proof: Let x be any point in V. Then $x \in U$ for some open set $U \in \mathcal{U}$. Since U is open, there exists a positive r such that $U_r(x) \subset U$, and since $U \subset V, U_r(x) \subset V$. □

Theorem A.2 *Let $\mathcal{U}$ be any finite collection of open sets of metric space $(\mathbb{X}, d)$. Then*

$$V = \bigcap_{U_i \in \mathcal{U}} U \tag{11}$$

is open.

Proof: Let x be any point in V. Then $x \in U_i$ for all $U_i \in \mathcal{U}$. Since each U_i is open, there exists a corresponding positive r_i such that $U_{r_i}(x) \subset U_i$. Let

$$r = \min_{i=1\ldots \mathrm{card}\mathcal{U}} r_i. \tag{12}$$

Then $U_r(x) \in V$. □

Definition A.4 *Let $x_1, x_2, \ldots$ be a sequence of points in metric space $(\mathbb{X}, d)$. The sequence* converges *to a* limit point *$x \in \mathbb{X}$ if and only if, for any $\epsilon > 0$ there exists an N large enough that for all $i > N$,*

$$d(x_i, x) < \epsilon. \tag{13}$$

Definition A.5 *A set A of metric space $(\mathbb{X}, d)$ is closed if and only if any sequence $x_1, x_2, \ldots$ of points $x_i \in A$ that converges to a point $x \in \mathbb{X}$, implies that $x \in A$.*

Theorem A.3 *Let A be a subset of $\mathbb{X}$ and let $U = \mathbb{X} \setminus A$. Then A is closed if and only if U is open.*

Proof: Let U be open and let $x_1, x_2, \ldots$ be any sequence in A converging to a point $x \in X$. If $x \in U$ then $U_r(x) \subset U$ for some positive r. But given an r implies an N such that $d(x_i, x) < r$ for some $i > N$, implying $x_i \in U$! Thus, $x \in A$ and A is closed.

Let A be closed and let x be a point in U such that $S_r(x) \not\subset U$ for any positive r. Then let

$$r_i = \frac{1}{i} \tag{14}$$

define a decreasing sequence of radii. For each r_i let x_i be any point in $A \cap U_{r_i}(x)$. Then x_i converges to x. But for each $i, x_i \in A$ which is closed, implying $x \in A$. Thus, every $x \in U$ is surrounded by an open ball in U and U is open. □

Corollary A.4 *Let $\mathcal{A}$ be any finite collection of closed sets of metric space $(\mathbb{X}, d)$. Then*

$$B = \bigcup_{A_i \in \mathcal{A}} A_i \tag{15}$$

is closed.

Corollary A.5 *Let $\mathcal{A}$ be any collection of closed sets of metric space $(\mathbb{X}, d)$. Then*

$$B = \bigcap_{A \in \mathcal{A}} A \tag{16}$$

is closed.

Definition A.6 *Let $(\mathbb{X}, d)$ be a metric space and let $x_1, x_2, \ldots$ be a sequence in $\mathbb{X}$ such that for any $\epsilon > 0$ there exists an $N > 0$ such that*

$$d(x_i, x_j) < \epsilon \tag{17}$$

for all $i, j > N$. Then $(\mathbb{X}, d)$ is a complete *metric space if and only if all such previously described sequences in $\mathbb{X}$ converge to a point in $\mathbb{X}$.*

Definition A.7 *Let A be a subset of metric space $(\mathbb{X}, d)$. Then A is* compact *if and only if given any collection of open sets $\mathcal{U}$ such that*

$$A \subset \bigcup_{U \in \mathcal{U}} U \tag{18}$$

there exists a finite sub-collection $\mathcal{V} \subset \mathcal{U}$ such that

$$A \subset \bigcup_{U \in \mathcal{V}} U. \tag{19}$$

Appendix B

DOCUMENTATION

This appendix documents the codes used to describe iterated function systems that modeled the objects illustrated in this dissertation. The codes are part of a parameter file interface to a rendering system developed on the AT&T Pixel Machine 964dX using a Sun 4 host. The "newmap" specification initializes a new transformation matrix to the unit matrix. The other commands simply post-multiply the implied transformation matrix to the current one.

B.1 The Extruded Sierpinski's Gasket

Any 2-D IFS may be extruded into 3-D by duplicating its transformations. The original set of transformations is scaled by one-half in the z-axis and then translated by one-half in the positive z direction. The duplicate set of transformations are likewise scaled in the z-axis but a translated in the negative z direction.

```
ifs esg {
        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0,-0.5
```

```
        newmap
        scale: 0.5,0.5,0.5
        translate: 0.5,0,-0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.5,-0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0,0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.5,0,0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.5,0.5
}
```

B.2 Natural Models

The natural models are examples of the power of linear fractals for simulating natural objects.

B.2.1 Grass

The grass model uses a twindragon curve to efficiently fill space. The blades of grass are images of a single central blade which is itself a small squashed version of the whole.

The value of 0.96 was determined visually to produce a full lawn. A lesser amount produces a sparse, spikey lawn, whereas an amount closer to one produces a lawn that appears too solid.

The grass IFS is specified so that it will fit snugly inside a sphere of radius three. The grass should be "mowed" with a final contraction in the y-axis.

```
ifs grass {
        newmap
        scale: 0.707,0.96,0.707
        rotate: y,45
        translate: -1,0,0

        newmap
        scale: 0.707,0.96,0.707
        rotate: y,45
        translate: 1,0,0

        newmap
        scale: 0.25,0.01,0.01
        rotate: z,90
        translate: 0,0.707,0
}

object grass {
```

```
        sphere: (0,0,0) 3
        color: 0,0.7,0.2,1
        ifs: grass
        scale: 4,0.1,4
}
```

B.2.2 Elm Tree

The elm tree is a trunk with four smaller elm trees sticking out of it. The trunk consists of three upright stretched elm trees and three inverted stretched elm trees — the foliage becomes the bark.

The use of color here is quite delicate. A lot of "branch" transformations must be applied to create a green color whereas any "trunk" transformation will immediately cause a large amount of brown to be added.

```
ifs elm {
        newmap
        rotate: y,90
        scale: 0.6,0.6,0.6
        rotate: x,40
        translate: 0,1,0
        color: 0,1,0,0.05

        newmap
        rotate: y,75
        scale: 0.55,0.55,0.55
        rotate: x,-50
        translate: 0,0.9,0
```

```
color: 0,0.9,0.2,0.05

newmap
rotate: y,105
scale: 0.5,0.5,0.5
rotate: z,60
translate: 0,0.95,0
color: 0.2,0.9,0,0.05

newmap
rotate: y,95
scale: 0.45,0.45,0.45
rotate: z,-35
translate: 0,0.8,0
color: 0.4,0.8,0.4,0.05

newmap
rotate: y,35
scale: 0.055,0.25,0.055
translate: 0,0.2,0
color: 0.4,0.3,0.1,1

newmap
rotate: y,175
scale: 0.055,0.25,0.055
translate: 0,0.35,0
color: 0.4,0.3,0.1,1
```

```
newmap
rotate: y,50
scale: 0.0525,0.25,0.0525
translate: 0,0.5,0
color: 0.4,0.3,0.1,1

newmap
rotate: y,255
scale: 0.08,-0.25,0.08
translate: 0,0.4,0
color: 0.4,0.3,0.1,1

newmap
rotate: y,145
scale: 0.065,-0.25,0.065
translate: 0,0.55,0
color: 0.4,0.3,0.1,1

newmap
rotate: y,305
scale: 0.0575,-0.25,0.0575
translate: 0,0.7,0
color: 0.4,0.3,0.1,1

newmap
rotate: y,210
```

```
        scale: 0.05,-0.25,0.05
        translate: 0,0.85,0
        color: 0.4,0.3,0.1,1
}
```

B.2.3 Pine Tree

The pine tree is much like a 3-D version of the spleenwort fern. It consists of six perturbed base-branch transformations and one "rest of the tree" transformation. Like the grass, it is designed to fit into a canonical cone, and should be transformed to better simulate the proportions of a pine tree.

```
ifs pine {
        newmap
        scale: 0.2,0.7,0.2
        rotate: z,96
        translate: 0,0.1,0
        color: 0,0.6,0,0.1

        newmap
        scale: 0.2,0.7,0.2
        rotate: z,90
        rotate: y,63
        translate: 0,0.1,0
        color: 0,0.6,0,0.1

        newmap
        scale: 0.2,0.7,0.2
```

```
rotate: z,93
rotate: y,104
translate: 0,0.1,0
color: 0,0.6,0,0.1

newmap
scale: 0.2,0.7,0.2
rotate: z,97
rotate: y,167
translate: 0,0.1,0
color: 0,0.6,0,0.1

newmap
scale: 0.2,0.7,0.2
rotate: z,95
rotate: y,253
translate: 0,0.1,0
color: 0,0.6,0,0.1

newmap
scale: 0.2,0.7,0.2
rotate: z,100
rotate: y,311
translate: 0,0.1,0
color: 0,0.6,0,0.1

newmap
```

```
        rotate: y,31
        scale: 0.82,0.84,0.82
        translate: 0,0.15,0
        color: 0,0.6,0.6,0.2
}

object pine {
        cone
        color: 1,0,0,0
        ifs: pine
        scale: 1.5,2,1.5
        translate: 1,0.1,0
}
```

B.3 The Five Non-Platonic Non-Solids

The five non-Platonic non-Solids are fractal versions of the five Platonic solids. They are simple examples of 3-D iterated function systems, consisting of maps whose fixed points are found at the shapes' vertices.

B.3.1 Sierpinski's Tetrahedron

Sierpinski's tetrahedron is a natural extension of Sierpinki's gasket. It is called a "skewed web" in [Mandelbrot, 1982b]. This object has been commonly used to test the effectiveness of rendering algorithms.

```
ifs tetra {
        newmap
```

```
        scale: 0.5,0.5,0.5
        translate: 0,0.5,0

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,-0.166,-0.472

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.409,-0.166,0.236

        newmap
        scale: 0.5,0.5,0.5
        translate: -0.409,-0.166,0.236
}
```

B.3.2 Sierpinski's Octahedron

Unlike Sierpinski's tetrahedron, Sierpinski's octohedron is complete opaque at all viewing angles.

```
ifs octo {
        newmap
        scale: 0.5,0.5,0.5
        translate: -0.5,0,0

        newmap
        scale: 0.5,0.5,0.5
```

```
        translate: 0.5,0,0

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0,-0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0,0.5

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.5,0

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,-0.5,0
}
```

B.3.3 Menger's Sponge

Perhaps a better analog to the hexahedron would be an IFS consisting of eight maps that take the cube to each of its octants. Unfortunately its attractor is not fractal. Hence, Menger's sponge is used as a fractal equivalent to the hexahedron.

```
ifs menger {
        newmap
        scale: 0.333,0.333,0.333
```

```
translate: 0.385,0.385,0.385

newmap
scale: 0.333,0.333,0.333
translate: -0.385,0.385,0.385

newmap
scale: 0.333,0.333,0.333
translate: 0.385,-0.385,0.385

newmap
scale: 0.333,0.333,0.333
translate: -0.385,-0.385,0.385

newmap
scale: 0.333,0.333,0.333
translate: 0.385,0.385,-0.385

newmap
scale: 0.333,0.333,0.333
translate: -0.385,0.385,-0.385

newmap
scale: 0.333,0.333,0.333
translate: 0.385,-0.385,-0.385

newmap
```

```
scale: 0.333,0.333,0.333
translate: -0.385,-0.385,-0.385

newmap
scale: 0.333,0.333,0.333
translate: 0.385,0,0.385

newmap
scale: 0.333,0.333,0.333
translate: -0.385,0,0.385

newmap
scale: 0.333,0.333,0.333
translate: 0.385,0,-0.385

newmap
scale: 0.333,0.333,0.333
translate: -0.385,0,-0.385

newmap
scale: 0.333,0.333,0.333
translate: 0,0.385,0.385

newmap
scale: 0.333,0.333,0.333
translate: 0,-0.385,0.385
```

```
        newmap
        scale: 0.333,0.333,0.333
        translate: 0,0.385,-0.385

        newmap
        scale: 0.333,0.333,0.333
        translate: 0,-0.385,-0.385

        newmap
        scale: 0.333,0.333,0.333
        translate: 0.385,0.385,0

        newmap
        scale: 0.333,0.333,0.333
        translate: -0.385,0.385,0

        newmap
        scale: 0.333,0.333,0.333
        translate: 0.385,-0.385,0

        newmap
        scale: 0.333,0.333,0.333
        translate: -0.385,-0.385,0
}
```

B.3.4 Sierpinski's Icosahedron

Sierpinski's icosahedron has small Sierpinski's gaskets on its faces.

```
ifs icos {
        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.926,0.263

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.926,-0.263

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.075,0.263

        newmap
        scale: 0.5,0.5,0.5
        translate: 0,0.075,-0.263

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.263,0.5,0.425

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.263,0.5,-0.425
```

```
        newmap
        scale: 0.5,0.5,0.5
        translate: -0.263,0.5,0.425

        newmap
        scale: 0.5,0.5,0.5
        translate: -0.263,0.5,-0.425

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.425,0.763,0

        newmap
        scale: 0.5,0.5,0.5
        translate: -0.425,0.763,0

        newmap
        scale: 0.5,0.5,0.5
        translate: 0.425,0.237,0

        newmap
        scale: 0.5,0.5,0.5
        translate: -0.425,0.237,0
}
```

B.3.5 Von Koch's Snowflake-a-hedron

Von Koch's snowflake-a-hedron is the fractal analog of the dodecahedron. Its profile is similar to the von Koch snowflake curve, hence the name. In [Hart, 1991], a geometric one-to-one correspondence was shown between its twenty maps and the twenty maps of Menger's sponge.

```
ifs dodec {
        newmap
        scale: 0.382,0.382,0.382
        translate: -0.577,0.618,-0.221

        newmap
        scale: 0.382,0.382,0.382
        translate: 0.577,0.618,-0.221

        newmap
        scale: 0.382,0.382,0.382
        translate: 0.577,0.618,0.221

        newmap
        scale: 0.382,0.382,0.382
        translate: -0.577,0.618,0.221

        newmap
        scale: 0.382,0.382,0.382
        translate: -0.357,0.975,-0.357

        newmap
```

```
scale: 0.382,0.382,0.382
translate: 0,0.839,-0.577

newmap
scale: 0.382,0.382,0.382
translate: 0.357,0.975,-0.357

newmap
scale: 0.382,0.382,0.382
translate: 0.221,1.195,0

newmap
scale: 0.382,0.382,0.382
translate: 0.357,0.975,0.357

newmap
scale: 0.382,0.382,0.382
translate: 0,0.839,0.577

newmap
scale: 0.382,0.382,0.382
translate: -0.357,0.975,0.357

newmap
scale: 0.382,0.382,0.382
translate: -0.221,1.195,0
```

```
newmap
scale: 0.382,0.382,0.382
translate: -0.357,0.261,-0.357

newmap
scale: 0.382,0.382,0.382
translate: 0,0.397,-0.577

newmap
scale: 0.382,0.382,0.382
translate: 0.357,0.261,-0.357

newmap
scale: 0.382,0.382,0.382
translate: 0.221,0.041,0

newmap
scale: 0.382,0.382,0.382
translate: 0.357,0.261,0.357

newmap
scale: 0.382,0.382,0.382
translate: 0,0.397,0.577

newmap
scale: 0.382,0.382,0.382
translate: -0.357,0.261,0.357
```

```
        newmap
        scale: 0.382,0.382,0.382
        translate: -0.221,0.041,0
}
```

CITED LITERATURE

[Abelson & diSessa, 1982] Abelson, H. and diSessa, A. A. *Turtle Geometry.* MIT Press, 1982.

[Amanatides & Mitchell, 1989] Amanatides, J. and Mitchell, D. P. Megacycles. *SIGGRAPH Video Review*, 51, 1989. (Animation).

[Amanatides, 1984] Amanatides, J. Ray tracing with cones. *Computer Graphics*, 18(3):129–135, July 1984.

[Appel, 1968] Appel, A. Some techniques for shading machine renderings of solids. *AFIPS 1968 Spring Joint Computer Conference*, 32:37–45, 1968.

[Barnsley & Demko, 1985] Barnsley, M. F. and Demko, S. G. Iterated function schemes and the global construction of fractals. *Proceedings of the Royal Society A*, 399:243–275, 1985.

[Barnsley *et al.*, 1986] Barnsley, M. F., Ervin, V., Hardin, D., and Lancaster, J. Solution of an inverse problem for fractals and other sets. *Proceedings of the National Academy of Science*, 83:1975–1977, April 1986.

[Barnsley *et al.*, 1988] Barnsley, M. F., Jacquin, A., Mallassenet, F., Rueter, L., and Sloan, A. D. Harnessing chaos for image synthesis. *Computer Graphics*, 22(4):131–140, 1988.

[Barnsley *et al.*, 1989] Barnsley, M. F., Elton, J. H., and Hardin, D. P. Recurrent iterated function systems. *Constructive Approximation*, 5:3–31, 1989.

[Barnsley, 1988] Barnsley, M. F. *Fractals Everywhere*. Academic Press, New York, 1988.

[Barr, 1984] Barr, A. H. Global and local deformations of solid primitives. *Computer Graphics*, 18(3):21–30, July 1984.

[Barr, 1986] Barr, A. H. Ray tracing deformed surfaces. *Computer Graphics*, 20(4):287–296, 1986.

[Bouville, 1985] Bouville, C. Bounding ellipsoids for ray-fractal intersection. *Computer Graphics*, 19(3):45–51, 1985.

[Cabrelli *et al.*, 1991] Cabrelli, C., Molter, U., and Vrscay, E. R. Recurrent iterated function systems: Invariant measures, a collage theorem and moment relations. In *Proceedings of the First IFIP Conference on Fractals*. Elsevier, 1991.

[Catmull, 1975] Catmull, E. Computer display of curved surfaces. In *IEEE Conf. on Computer Graphics, Pattern Recognition and Data Structures*, pages 11–17, May 1975.

[Catmull, 1978] Catmull, E. A hidden-surface algorithm with anti-aliasing. *Computer Graphics*, 12(3):275–281, August 1978.

[Cook *et al.*, 1984] Cook, R. L., Porter, T., and Carpenter, L. Distributed ray tracing. *Computer Graphics*, 18(3):137–145, 1984.

[Cook, 1986] Cook, R. L. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[Dekking, 1982] Dekking, F. M. Recurrent sets. *Advances in Mathematics*, 44:78–104, 1982.

[Demko *et al.*, 1985] Demko, S., Hodges, L., and Naylor, B. Construction of fractal objects with iterated function systems. *Computer Graphics*, 19(3):271–278, 1985.

[Deo, 1972] Deo, N. *Graph Theory with Applications to Engineering and Computer Science.* Prentice Hall, Englewood Cliffs, N.J., 1972.

[Dubuc & Elqortobi, 1990] Dubuc, S. and Elqortobi, A. Approximations of fractal sets. *Journal of Computational and Applied Mathematics*, 29:79–89, 1990.

[Falconer, 1985] Falconer, K. J. *The Geometry of Fractal Sets.* Cambridge University Press, New York, 1985.

[Falconer, 1987] Falconer, K. J. The Hausdorff dimension of some fractals and attractors of overlapping construction. *Journal of Statistical Physics*, 47((1&2)):123–132, 1987.

[Falconer, 1988] Falconer, K. J. The Hausdorff dimension of self-affine fractals. *Proceedings of the Cambridge Philosophical Society*, 103:339–350, 1988.

[Falconer, 1990] Falconer, K. J. *Fractal Geometry: Mathematical Foundations and Applications.* John Wiley & Sons, New York, 1990.

[Foley *et al.*, 1990] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. *Computer Graphics: Principles and Practice.* Systems Programming Series. Addison-Wesley, 2nd edition, 1990.

[Fournier *et al.*, 1982] Fournier, A., Fussel, D., and Carpenter, L. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, June 1982.

[Gardner, 1984] Gardner, G. Y. Simulation of natural scenes using textured quadric surfaces. *Computer Graphics*, 18(3):11–20, July 1984.

[Hanrahan, 1983] Hanrahan, P. Ray tracing algebraic surfaces. *Computer Graphics*, 17(3):83–90, 1983.

[Hart & DeFanti, 1991] Hart, J. C. and DeFanti, T. A. Efficient antialiased rendering of 3-D linear fractals. *Computer Graphics*, 25(3), 1991.

[Hart *et al.*, 1989] Hart, J. C., Sandin, D. J., and Kauffman, L. H. Ray tracing deterministic 3-D fractals. *Computer Graphics*, 23(3):289–296, 1989.

[Hart *et al.*, 1990] Hart, J. C., Kauffman, L. H., and Sandin, D. J. Interactive visualization of quaternion Julia sets. In Kauffman, A., editor, *Proceedings of Visualization '90*, pages 209–218. IEEE Computer Society, 1990.

[Hart, 1989] Hart, J. C. Image space algorithms for visualizing quaternion Julia sets. Master's thesis, EECS Dept., University of Illinois at Chicago, 1989.

[Hart, 1991] Hart, J. C. unNatural Phenomena. *SIGGRAPH Video Review*, 71, 1991. (Animation).

[Heckbert & Hanrahan, 1984] Heckbert, P. S. and Hanrahan, P. Beam tracing polygonal objects. *Computer Graphics*, 18(3):119–127, July 1984.

[Hepting *et al.*, 1990] Hepting, D., Prusinkiewicz, P., and Saupe, D. Rendering methods for iterated function systems. In *Proceedings of Fractals '90*. IFIP, 1990.

[Hepting, 1991] Hepting, D. H. Approximation and visualization of sets defined by iterated function systems. Master's thesis, University of Regina, 1991.

[Holbrook, 1983] Holbrook, J. A. R. Quaternionic astroids and starfields. *Applied Mathematical Notes*, 8(2):1–34, 1983.

[Holbrook, 1987] Holbrook, J. A. R. Quaternionic Fatou-Julia sets. *Annals of Science and Math Quebec*, 1987(1):79–94, 1987.

[Hutchinson, 1981] Hutchinson, J. Fractals and self-similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[Jaquin, 1991] Jaquin, A. E. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Signal Processing*, page (to appear), March 1991.

[Kajiya & Kay, 1989] Kajiya, J. T. and Kay, T. L. Rendering fur with three dimensional textures. *Computer Graphics*, 23(3):271–280, July 1989.

[Kajiya, 1982] Kajiya, J. T. Ray tracing parametric patches. *Computer Graphics*, 16(3):245–254, July 1982.

[Kajiya, 1983] Kajiya, J. T. New techniques for ray tracing procedurally defined objects. *ACM Transactions on Graphics*, 2(3):161–181, 1983. Also appeared in *Computer Graphics 17,* 3 (1983), 91–102.

[Kalra & Barr, 1989] Kalra, D. and Barr, A. H. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, 23(3):297–306, July 1989.

[Kaplansky, 1977] Kaplansky, I. *Set Theory and Metric Spaces.* Chelsea, New York, 1977.

[Kay & Kajiya, 1986] Kay, T. L. and Kajiya, J. T. Ray tracing complex scenes. *Computer Graphics*, 20(4):269–278, 1986.

[Lescinski, 1991] Lescinski, G. Lively ifs. *SIGGRAPH Video Review*, 61, 1991. (Animation).

[Mandelbrot & Ness, 1968] Mandelbrot, B. B. and Ness, J. W. V. Fractional brownian motions, fractional noise and applications. *SIAM Review*, 10(4):422–437, October 1968.

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

[Mandelbrot, 1982a] Mandelbrot, B. B. Comment on computer rendering of stochastic models. *Communications of the ACM*, 25(8):581–583, 1982.

[Mandelbrot, 1982b] Mandelbrot, B. B. *The Fractal Geometry of Nature*. W.H. Freeman, San Francisco, 2nd edition, 1982.

[Miller, 1986] Miller, G. S. P. The definition and rendering of terrain maps. *Computer Graphics*, 20(4):39–48, August 1986.

[Milnor, 1991] Milnor, J. Remarks on iterated cubic maps. In Hart, J. C. and Musgrave, F. K., editors, *Fractal Models in 3-D Computer Graphics and Imaging*, pages 193–221. ACM SIGGRAPH '91 (Course #14 Notes), 1991.

[Mitchell, 1987] Mitchell, D. P. Generating antialiased images at low sampling resolutions. *Computer Graphics*, 21(4):65–72, July 1987.

[Musgrave *et al.*, 1989] Musgrave, F. K., Kolb, C. E., and Mace, R. S. The synthesis and rendering of eroded fractal terrains. *Computer Graphics*, 23(3):41–50, July 1989.

[Nishita *et al.*, 1990] Nishita, T., Sederberg, T. W., and Kakimoto, M. Ray tracing trimmed rational surface patches. *Computer Graphics*, 24(4):337–345, Aug. 1990.

[Norton & Melton, 1988] Norton, A. and Melton, E. A close encounter in the fourth dimension. *SIGGRAPH Video Review*, 39, 1988. (Animation).

[Norton *et al.*, 1982] Norton, A., Rockwood, A. P., and Skolmoski, P. T. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. *Computer Graphics*, 16(3):1–8, July 1982.

[Norton, 1982] Norton, A. Generation and rendering of geometric fractals in 3-D. *Computer Graphics*, 16(3):61–67, 1982.

[Norton, 1989a] Norton, A. Fractal transitions. *SIGGRAPH Video Review*, 42, 1989. (Animation).

[Norton, 1989b] Norton, A. Julia sets in the quaternions. *Computers and Graphics*, 13(2):267–278, 1989.

[Peitgen *et al.*, 1991] Peitgen, H.-O., Jurgens, H., and Saupe, D. *Fractals for the Classroom.* Springer-Verlag, New York, 1991.

[Penrose, 1989] Penrose, R. *The Emperor's New Mind.* Harcourt-Brace, 1989.

[Perlin, 1985] Perlin, K. An image synthesizer. *Computer Graphics*, 19(3):287–296, July 1985.

[Porter & Duff, 1984] Porter, T. and Duff, T. Compositing digital images. *Computer Graphics*, 18(3):253–259, 1984.

[Potmesil & Hoffert, 1989] Potmesil, M. and Hoffert, E. M. The Pixel Machine: A parallel image computer. *Computer Graphics*, 23(3):69–78, July 1989.

[Preparata & Shamos, 1985] Preparata, F. P. and Shamos, M. I. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[Press *et al.*, 1988] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C.* Cambridge University Press, 1988.

[Prusinkiewicz & Hammel, 1991] Prusinkiewicz, P. and Hammel, M. Automata, languages and iterated function systems. In Hart, J. C. and Musgrave, F. K., editors, *Fractal Models in 3-D Computer Graphics and Imaging*, pages 115–143. ACM SIGGRAPH '91 (Course #14 Notes), 1991.

[Prusinkiewicz & Hanan, 1989] Prusinkiewicz, P. and Hanan, J. *Lindenmeyer Systems, Fractals, and Plants*, volume 79 of *Lecture Notes in Biomathematics.* Springer-Verlag, New York, 1989.

[Prusinkiewicz & Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[Prusinkiewicz *et al.*, 1988] Prusinkiewicz, P., Lindenmayer, A., and Hanan, J. Developmental models of herbaceous plants for computer imagery purposes. *Computer Graphics*, 22(4):141–150, August 1988.

[Reuter, 1987] Reuter, L. *Rendering and Magnification of Fractals Using Iterated Function Systems*. PhD thesis, Georgia Institute of Technology, December 1987.

[Ritter, 1990] Ritter, J. An efficient bounding sphere. In Glassner, A. S., editor, *Graphics Gems*, pages 301–303. Academic Press, 1990.

[Roth, 1982] Roth, S. D. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, February 1982.

[Rubin & Whitted, 1980] Rubin, S. M. and Whitted, T. A 3-dimensional representation for fast rendering of complex scenes. *Computer Graphics*, 14(3):110–116, 1980.

[Sandin *et al.*, 1990] Sandin, D. J., Hart, J. C., and Kauffman, L. H. Interactive visualization of complex, stacked and quaternion Julia sets. In *Proceedings of Ausgraph '90*, 1990.

[Shannon, 1949] Shannon, C. E. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37(1):10–21, January 1949.

[Shroeder, 1991] Shroeder, M. R. *Fractals, Chaos, Power Laws*. W.H. Freeman, San Francisco, 1991.

[Smith, 1984] Smith, A. R. Plants, fractals, and formal languages. *Computer Graphics*, 18(3):1–10, July 1984.

[Snyder & Barr, 1987] Snyder, J. M. and Barr, A. H. Ray tracing complex models containing surface tessellations. *Computer Graphics*, 21(4):119–128, 1987.

[Strang, 1988] Strang, G. *Linear Algebra and its Applications*. Harcourt Brace Jovanovich, 3rd edition, 1988.

[Sutherland *et al.*, 1974] Sutherland, I. E., Sproul, R., and Schumacker, R. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1):1–55, 1974.

[Sutherland, 1963] Sutherland, I. E. Sketchpad: A man-machine graphical communication system. *Proceedings of the Spring Joint Computer Conference*, 1963.

[Thomas *et al.*, 1989] Thomas, D., Netravali, A. N., and Fox, D. S. Antialiased ray tracing with covers. *Computer Graphics Forum*, 8(4):325–336, December 1989.

[Thompson, 1991] Thompson, K. K. *Ray Tracing with Amalgams*. PhD thesis, University of Texas at Austin, 1991.

[Toth, 1985] Toth, D. L. On ray tracing parametric surfaces. *Computer Graphics*, 19(3):171–179, July 1985.

[Voss, 1988] Voss, R. F. Fractals in nature: From characterization to simulation. In Peitgen, H. and Saupe, D., editors, *The Science of Fractal Images*, pages 21–70. Springer-Verlag, New York, 1988.

[Vrscay & Roehrig, 1989] Vrscay, E. R. and Roehrig, C. J. Iterated function systems and the inverse problem of fractal construction using moments. In Kaltofen, E. and Watt, S. M., editors, *Computers and Mathematics*, pages 250–259, New York, 1989. Springer-Verlag.

[Vrscay, 1991] Vrscay, E. R. Iterated function systems: Theory, applications and the inverse problem. In *Lectures of the NATO Advanced Study Institute on Fractal Geometry and Analysis*, Montreal, 1991. Kluwer.

[Whitted, 1980] Whitted, T. An improved illumination model for shaded displays. *Communications of the ACM*, 23(6):343–349, 1980.

[Womack, 1989] Womack, T. E. Linear and markov iterated function systems in fractal geometry. Master's thesis, Virginia Polytechnic Institute, May 1989.

VITA

Name: John C. Hart

Education: B.S., Computer Science, Aurora University, Aurora, Illinois, 1987.

M.S., Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, Illinois, 1989.

Ph.D., Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, Illinois, 1991.

Research: Postdoctoral Research Associate — Electronic Visualization Laboratory, University of Illinois at Chicago, Chicago, Illinois and National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1991–1992.

Research Assistant — Electronic Visualization Laboratory, University of Illinois at Chicago, Chicago, Illinois, 1987–1991.

Intern — AT&T Pixel Machines, AT&T Bell Laboratories, Holmdel, New Jersey, Summer 1990.

Intern — IBM T.J. Watson Research Center, Hawthorne, New York, Summer 1989.

Teaching: Fractal Models in 3-D Computer Graphics and Imaging — SIGGRAPH '91 One-day course #14, Caesar's Palace, Las Vegas, Nevada, 1991.

Scientific Visualization by Supercomputer — Adler Planetarium evening course, Chicago, Illinois, 1988.

Teaching Assistant — Electrical Engineering and Computer Science Department, University of Illinois at Chicago, Chicago, Illinois, 1987–1988.

Honors: Student Fellowship — Graduate College, University of Illinois at Chicago, Chicago, Illinois, 1990–1991.

Activities: Supercomputing '92 — Video Chair, Programming Committee.

SIGGRAPH '92 — Electronic Theater Committee, Chair Care Committee.

Supercomputing '91 — Programming Committee.

SIGGRAPH '91 — Course Organizer.

Publications: John C. Hart and F. Kenton Musgrave, Editors. Fractal Modeling in 3-D Computer Graphics and Imaging. SIGGRAPH '91 Course Notes, 1991.

John C. Hart and Thomas A. DeFanti. Efficient antialiased rendering of 3-D linear fractals. *Computer Graphics*, 25(3), 1991.

John C. Hart, Louis H. Kauffman, and Daniel J. Sandin. Interactive visualization of quaternion Julia sets. In Arie Kauffman, editor, *Proceedings of Visualization '90*, pages 209–218. IEEE Computer Society, 1990.

John C. Hart, Daniel J. Sandin, and Louis H. Kauffman. Ray tracing deterministic 3-D fractals. *Computer Graphics*, 23(3):289–296, 1989.

Animations: unNatural Phenomena. SIGGRAPH Video Review 71, 1991.

Sierpinski Blows His Gasket. SIGGRAPH Video Review 61, 1990.

Dynamics in the Quaternions. SIGGRAPH Video Review 42, 1989.