AN ABSTRACT OF THE THESIS OF

Jerry Wayne Hosack for the Master of Science

in Mathematics presented on August 22, 1985

Title: Fractals, Coastlines, and Computer Recreations

Abstract approved: Marion P. Emerson

This thesis will introduce the reader to some of the basic ideas and concepts involved in the study of fractal geometry and will apply these ideas to coastlines. Finally, the paper will use these ideas to generate some of the fractal patterns on the microcomputer.

FRACTALS, COASTLINES, AND COMPUTER RECREATIONS

A Thesis
submitted to
The Division of Mathematical and Physical Sciences
Emporia State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By
Jerry Wayne Hosack
August, 1985

Marion P. Emerson

Approved for the Major Department

Harold E Durst

Approved for the Graduate Council

TABLE OF CONTENTS

Chapter I

Introduction

Fractal geometry is a new approach to an old category of problems which was pioneered by Benoit B. Mandelbrot, a mathematician at IBM. Within the past decade, Mandelbrot's ideas have been gaining more and more support as new applications in a number of the natural sciences are investigated.

This paper will introduce the reader to some of the basic ideas and concepts involved in the study of fractal geometry and will apply these ideas to coastlines. Finally, the paper will use these ideas to generate some of the fractal patterns on the microcomputer. At the present time there are few reference books on this subject, and no text books are available. The material presented in Mandelbrot's The Fractal Geometry of Nature (1982) is basically an updated version of his 1977 book, Fractals, Form, Chance and Dimension. The former serves as the primary motivation for this paper.

FRACTALS IN NATURE

A fractal may be thought of as a shape whose parts are similar to each other and have the same degree of complexity on many different scales. Many shapes found in nature may be described more easily in terms of fractal shapes than in terms of standard shapes found in geometry. If a portion of a circle or a regular polygon is magnified over and over, the resulting image tends to straighten out. The earth may be used as an analogy. It appears to be flat because we see only a small

portion on a large scale. The surface of tree bark is very rough and crinkled. When tree bark is magnified, the resulting surface is also rough and crinkled. Consider the jagged line representing the vertical cross section of a piece of tree bark. When a portion of the cross section is enlarged, the result is another crinkled line which represents that portion in greater detail. A very jagged line may be used only as a first approximation to the vertical cross section of a piece of tree bark because when the jagged line is magnified, it straightens out in the same manner as for a circle. A map of an island may be enlarged over and over to produce maps of finer and finer detail; however, the coastline of the island generally does not straighten with magnification. If a set of maps of a small island were drawn to many different scales, each map would reveal different detail, but the overall jagged coastal pattern would be the same.

The human lung is another shape which is hard for standard geometry to describe because it is made up of a complex of tubes, air sacks, and blood vessels of many different scales. A cloud's shape is determined by a collection of many tiny particles which make up pillow shapes which make up larger pillow shapes which make up still larger pillow shapes. Mandelbrot writes, "I claim that many patterns of Nature are so irregular and fragmented, that, compared with Euclid--a term used ... to denote all of standard geometry, Nature exhibits not simply a higher degree but an altogether different level of complexity. The number of distinct scales

of length of natural patterns is for all practical purposes infinite" [5, page 1].

EARLY BEGINNINGS AND APPLICATIONS OF FRACTAL GEOMETRY

The word fractal was coined by Benoit B. Mandelbrot from the Latin adjective *fractus*, meaning irregular, and the Latin verb *frangere*, which means to break or create irregular fragments. The term "fractal" first appeared in Mandelbrot's 1977 book.

One of the first applications which led to the development of fractal geometry was introduced in 1958 when Mandelbrot convinced IBM to halt a multimillion-dollar research project aimed at eliminating noise from their systems. Mandelbrot showed that noise, i.e., random fluctuations in signal transmission, is impossible to eliminate entirely. He described the noise using strange new logic.

Later, around 1968, Mandelbrot was working on what he called the "Joseph effect," a term used to describe the yearly water levels of the Nile river. The fluctuations in the water levels are very persistent, and records indicate that particular water levels have lasted for as long as a thousand years. Mandelbrot says, "If you look at a record of the Nile's discharges, you don't see little flags that mark the beginning or the end of a drought. Each record seems to look like random noise superimposed on a background that is also noisy. The background seems cyclic, but you can't extrapolate from its cycles for predictive purposes. They are not periodic" [3, page 102]. Because of these features, the Nile

only remotely fits statistical and hydrology textbook models. Mandelbrot devised a fractal model based on data collected by the noted niologist Harold Edwin Hurst. He showed the graphs generated with his model, along with graphs made from the Hurst data, to a panel of hydrologists. They were unable to distinguish his forgeries from the authentic records of the Nile.

Mandelbrot also used fractals to model stock market prices. His artificial graphs of cotton prices were mixed with authentic graphs as well as graphs produced by other computer models. The collection was taken to a prominent stockbroker. The stockbroker identified the charts produced by the standard models but was unable to distinguish the real records from Mandelbrot's artificial ones.

Much of Mandelbrot's earlier work dealt with computer generated coastlines. He was able to model coastlines which remarkably resembled actual coasts with a single set of equations. By changing a single parameter, the model was able to generate a large spectrum of coastlines from smooth, elongated islands like Taiwan to a very complex archipelago of shapes like the islands found in the Aegean Sea. Later, the methods were extended with the contributions of Richard Voss, a physicist also at IBM, to generating fractal mountains and landscapes. The parameter Mandelbrot used in these models is called the "fractal dimension" and is giving scientists a way to describe a complex phenomenon with a single number [3, page 65].

Fractal shapes and patterns are being recognized in many different natural sciences. For example, the frequencies of word usage in different languages are largely the same. This phenomenon, known as the "Ziph Law," is now being understood through a fractal approach to analysis.

Fractals are also being used in modeling the vegetation patterns of the Okefenokee Swamp in Georgia. The outlines of different groves or patches of trees look like coastlines when observed from the air. Harold Hastings, Professor of mathematics at Hofstra University on Long Island, claims that certain trees, such as cypress, are patchier than others and may be described in fractal terms. Noting changes in fractal patterns may serve as an early-warning system for acid rain or other harmful types of pollution [6, page 116].

In genetics, fractals suggest that a small amount of genetic code may be responsible for the growth of a large and complex organism, and minor changes in the code may result in global changes to the organism. Fractals are being used to write music, and the movie industry has invested heavily in computer equipment to generate fractal images. Other areas in which fractal analysis is being applied include the clustering and distribution of stellar matter, turbulence, distribution of oil and other natural resources, cratering of the moon, geometry of polymers, turbulence, chaos, the occurrence of earthquakes, the surfaces of metals, and meteorology.

Chapter II

Dimension

The concept of dimension plays a vital role in the study of fractals. Dimension is an elusive notion which historically has been used only in a vague sense. There has been no attempt made here to present a complete history of dimension, for indeed it would be quite lengthy. However, several discoveries made near the turn of the century have altered our previous ideas about geometry and dimension. Hurewicz and Wallman in their book Dimension Theory (1941) write, "The lack of a precise definition of dimension, however unsatisfactory from an esthetic and methodological point of view, caused no difficulty so long as geometry was confined to the study of relatively simple figures, such as polyhedra and manifolds. No doubt could arise, in each particular case as to what dimension to assign to each of these figures. This situation changed radically, following the discoveries of Cantor, with the development of point-set theory. This new branch of mathematics tremendously enlarged the domain of what could be considered as geometrical objects and revealed complexity never before dreamt of" [4, page 6].

Cantor showed a one-to-one correspondence between the points of a line and the points in a plane proving that dimension can be changed by a one-to-one transformation and that a plane does not contain more points than a line. Peano put forth a continuous mapping of an interval onto the whole of a square, a notion which contradicted the belief that the

dimension of a space could be defined as the least number of continuous real parameters required to describe the space, and showed that dimension can be raised by a one-valued continuous transformation [4, page 4]. Mandelbrot recognized that the graphs of such functions may be described, or "tamed," through a new approach he called fractal geometry.

Intuitive Dimension

There are many ways to define dimension on intuitive grounds. We deal with dimension each time we measure an object. We use units of measure which correspond to the dimension of the object we are trying to measure. For example, the length of a line may be given in feet (ft), the area of a region in square feet (ft^2), and the volume of a solid in cubic feet (ft^3). The exponent in each case is considered to be the dimension.

A more concise, but still intuitive, definition was written by Poincare in 1912 and serves as the introduction to Dimension Theory by Witold Hurewicz and Henry Wallman: "... if to divide a continuum it suffices to consider as cuts a certain number of elements all distinguishable from one another, we say that this continuum is of one dimension; if, on the contrary, to divide a continuum it is necessary to consider as cuts a system of elements themselves forming one or several continua, we shall say that this continuum is of several dimensions.

"If to divide a continuum C, cuts which form one or several continua of one dimension suffice, we shall say that C

is a continuum of two dimensions; if cuts which form one or several continua of at most two dimensions suffice, we shall say that C is a continuum of three dimensions; and so on" [4, page 3].

To give this idea a sense of validity, Poincare continues by comparing the above definition to that of geometers. "Usually they begin by defining surfaces as the boundaries of solids or pieces of space, lines as the boundaries of surfaces, points as the boundaries of lines, and they state that the same procedure can not be carried further.

"This is just the idea given above: to divide space, cuts that are called surfaces are necessary; to divide surfaces, cuts that are called lines are necessary; to divide lines, cuts that are called points are necessary; we can go no further and a point can not be divided, a point not being a continuum. Then lines, which can be divided by cuts which are not continua, will be continua of one dimension; surfaces, which can be divided by continuous cuts of one dimension, will be continua of two dimensions; and finally space, which can be divided by continuous cuts of two dimensions, will be a continuum of three dimensions" [4, page 3]. Euclidean geometry is based on these concepts and the mathematics of their properties, measurement, and relationships.

By looking at a fractal shape, the problems involved in measuring fractals as opposed to measuring shapes found in Euclidean geometry can be more easily understood.

The Triadic Koch Curve

This curve, otherwise known as the Snowflake curve, was first put forth by the German mathematician Helge Von Koch in 1904. Mandelbrot recognized this Koch curve as being a classic example of a fractal. The curve is constructed in stages, more detail being added with each successive stage. In generating fractal curves of this type, one begins with an N sided poloygon called an initiator. Then each straight interval is replaced by a shape called a generator. For the triadic Koch curve the initiator (figure 1) is a triangle, N = 3, with each side having length r1. The generator (figure 2) is a broken line made up of four line segments, each having length r2, where r2 is scaled to 1/3 the size of r1.

Initiator :

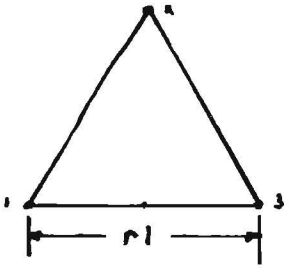


figure 1

generator :

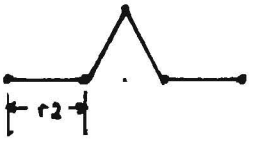


figure 2

The process proceeds by replacing the line segment between the first and second points of the initiator by the generator as shown in figure 3. Next, the line segments between the second and third points and between the third and first points are replaced by the generator. The second stage is now complete and shown in figure 4. To continue the construction, we

renumber the points and again replace the line segment between the first and second points with the generator, except scaled to 1/3 the size of r2, or (1/9) r1, as in figure 5. Next, the line segment between the second and third points is replaced, and so on, until all the line segments have been replaced. After the third stage the curve looks like that shown in figure 6.

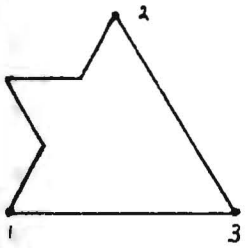


figure 3

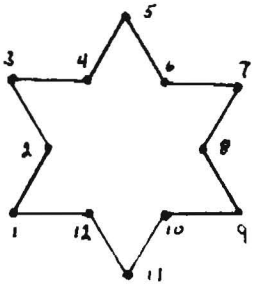


figure 4

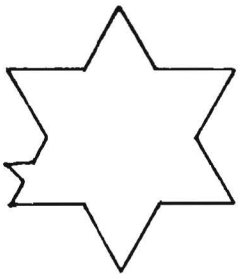


figure 5

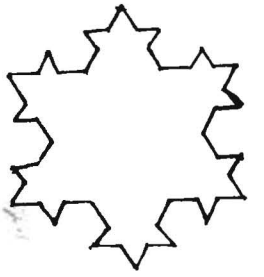


figure 6

This process is repeated again and again, stage after stage ad infinitum, replacing each interval with a scaled generator. When generating this curve on the computer, we are restricted by the resolution and the amount of memory available. So even with the most powerful equipment, the length of the resulting curve becomes only a finite approximation. However, theoretically, when one imagines this process being repeated over and over without bound, the curve becomes infinitely long. However,the area contained by the curve is finite. For example, given the initiator with N sides equal to 3, each having length r1 = 1, we have length L = N r1 = 3. For the second stage, N = 12 and r2 = 1/3; hence,

L = 4. For the third stage, N = 48 and r3 = 1/9; hence, L = 5 1/3 and so on. The length may be represented by the following geometric sequence where n denotes the stage of construction:

$$3,\quad 4,\quad 5\ 1/3,\ \ldots,\ L(n) \quad \text{where} \quad L(n) = 4\,(4/3)^{n-2}.$$

since $|r| = 4/3 > 1$; $\lim_{n\to\infty} L(n) = \infty$.

In the construction of the triadic Koch curve, triangular chunks of additional area are added to the initiator with each stage; however, the additional chunks become smaller and smaller with each successive stage. For example, let the area contained by the initiator be A = 1 (figure 7). The additional area obtained in the second stage (figure 8) is 3/9 since each of the three new triangles are 1/9 the size of the initiator. The third stage (figure 9) adds twelve more triangles, each 1/9 the size of its predecessor or 1/81 the size of the initiator. Hence, the actual area added with this stage is 12/81.

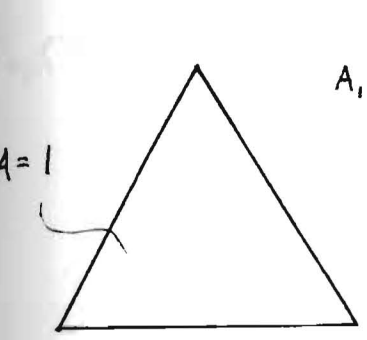


figure 7

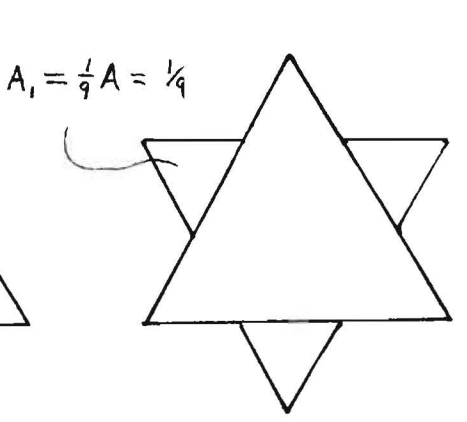


figure 8

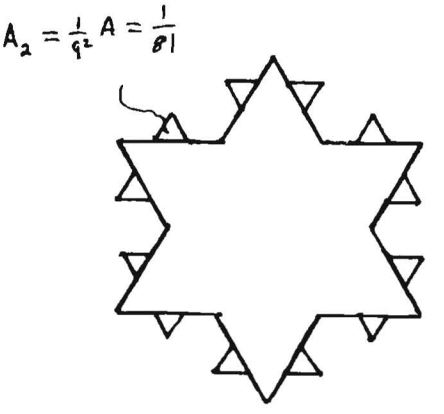


figure 9

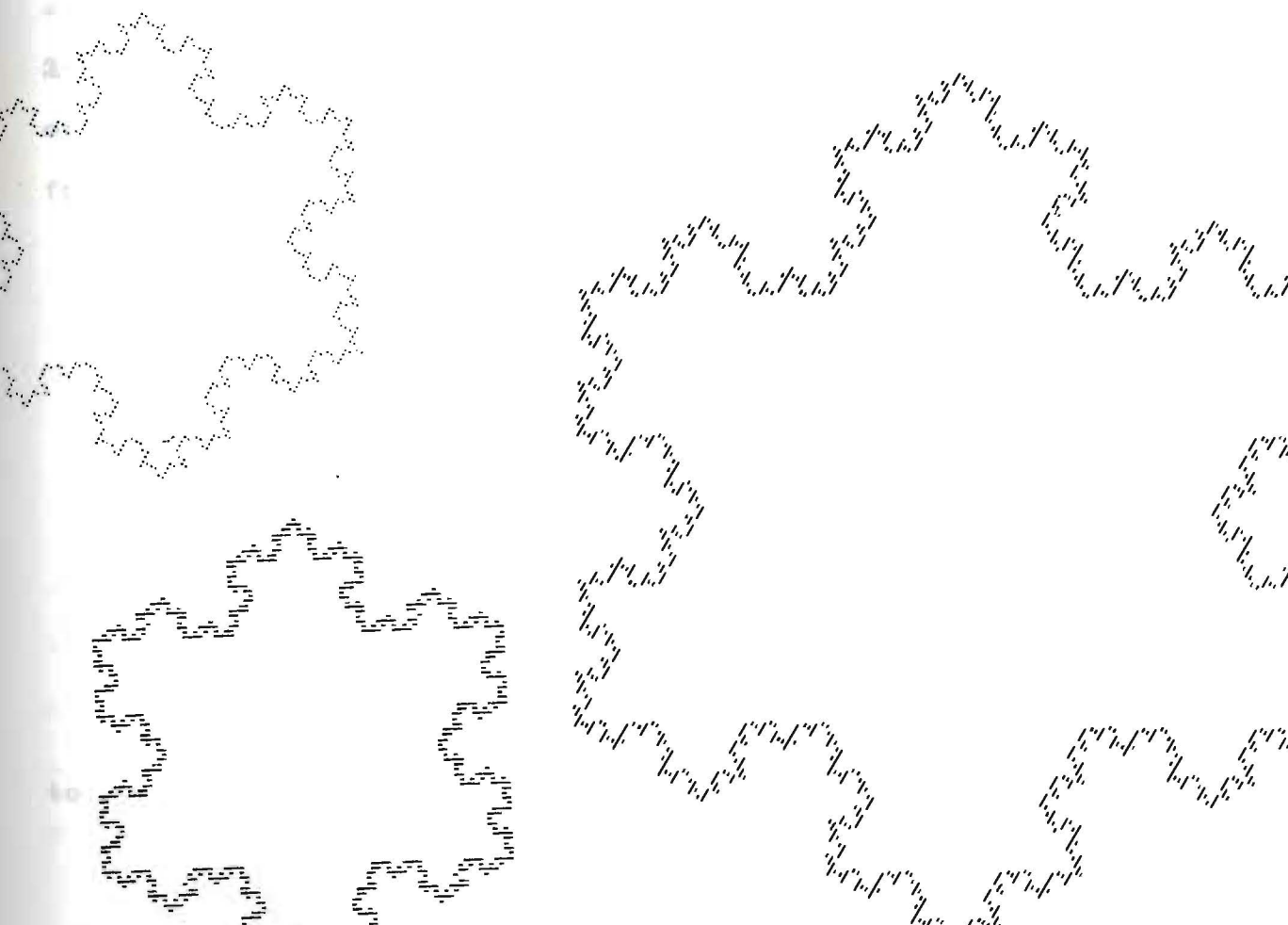
The area may be represented by the following geometric series:

Area = 1 + A(n)

where $A(n) = 3/9 + 12/9^{2} + 48/9^{3} + \ldots + 3/9\,(4/9)^{n-1}$.

Since $|r| = 4/9 < 1$, the series converges. Thus, the total area is given by $A = 1 + \lim_{n\to\infty} A(n) = 1 + (3/9) / (9/5) = 1.6$.

Since the curve enclosing this finite area is infinitely long, the length of any portion of the curve is infinite. Thus, a problem with using standard ideas about dimension and geometry as applied to the Koch curve becomes apparent when we try to compare the length of one portion of the curve to the length of another portion or to the whole. Shown below are some advanced stages of the triadic Koch curve.



The Problem With Measuring Fractal Shapes

A method in Euclidean geometry used to measure the length of a continuous curve F is to partition the curve into smaller lengths. Let P_0 , P_1 , ... , P_n be n + 1 points in 2-space. the curve R passing through these points is defined as follows:

$R(i) = P_i$, $i = 0$, ... , n ; if $i < t < i + 1$, then $R(t) = (1 - t + i)\, P_i + (t - i)\, P_{i+1}$. R is called the polygonal curve with vertices P_0 , P_1 , ... , P_n and is shown in figure 10. The length of polygonal curve P_0 , P_1 , ... , P_n is defined to be $\sum_{i=0}^{n-1} |P_i P_{i+1}|$, where $|P_i P_{i+1}|$ is the distance between points P_i and P_{i+1} . Let $P = (t_0, t_1, \ldots, t_n)$ be any partition of [a, b], that is, $a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b$, and let F be a curve in 2-space. The length of F is the least upper bound (supremum) of the lengths of the polygonal curves $R(t_0)\ R(t_1)\ \ldots\ R(t_n)$ for all partitions P of [a, b] (figure 11). [2, page 7]

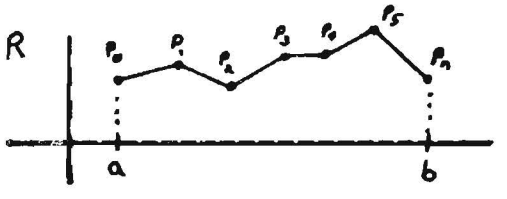


figure 10

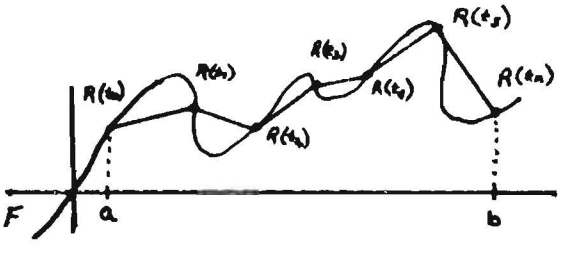


figure 11

Calculus extends these ideas and gives us powerful tools to use on a large group of shapes provided they are continuous

and have a first derivative. For example, notice the infinitesimal right triangle (figure 12) with legs having lengths |dx| and |dy| and hypotenise ds. By the Pathagorean theorem, $(ds)^2 = (dx)^2 + (dy)^2$.

Then, $ds = \left[1 + \left(\frac{dy}{dx}\right)^2\right]^{1/2} dx = \sqrt{1 + [f'(x)]^2}\, dx.$

Intergration of the differential gives the arc length of the graph between points a and b.

Hence, $S = \int_a^b ds = \int_a^b \sqrt{1 + [f'(x)]^2}\, dx$

[7, page 356].

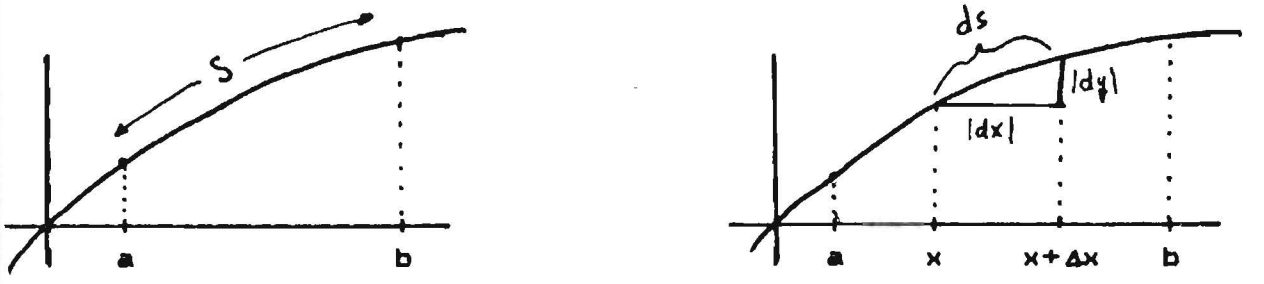


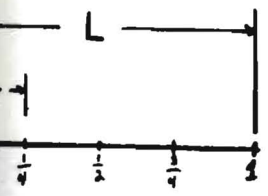
figure 12

If we try to measure the length of the Koch curve using either of these methods, little or no information is gained. The length determined by the first method is infinite. The second method cannot even be applied since the Koch curve has no derivative. Recall that the derivative of a function at a particular point exists if and only if both the one-sided

derivatives, i.e., from the left and right, exist, and if they all have the same value. Geometrically, the graph is differentiable at a point if it has a tangent line at that point.

The Koch curve has three points in the first stage of construction (figure 1) for which no tangent is defined. In the second stage (figure 4) there are twelve such points; in the third stage (figure 6) there are 48, and so on ad infinitum. It soon becomes evident that, because of the Koch curve's infinitely spiked nature, the derivative is undefined at every point on the curve. For these and other reasons this shape and ones like it were considered to be monstrous and pathological. Other than for purely theoretical reasons, very little attention has been given these shapes.

Consider the concepts of unit length, area, and volume using a standard line, square, and cube. The line can be broken into b equal parts, each having length 1/b. Likewise, the unit square can be broken into b^2 equal squares with the sides of each smaller square having length 1/b. The unit cube can be broken into b^3 equal cubes with the sides of each smaller cube having length 1/b. The exponent on b corresponds to the appropriate dimension d for a particular shape. Each part is deduced from the whole by a similarity, i.e., of ratio $r(N) = 1 / b = 1 / N^{1/d}$. It makes sense, then, to measure length in units of feet, area in square feet, and volume in cubic feet. This process is shown in figure 13.

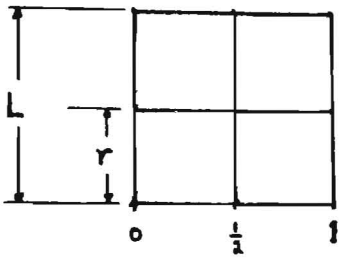


line: (length)

$n = b^1 = 4$

$r = 1/b = 1/4$

$L = n\,r = 1$ ft

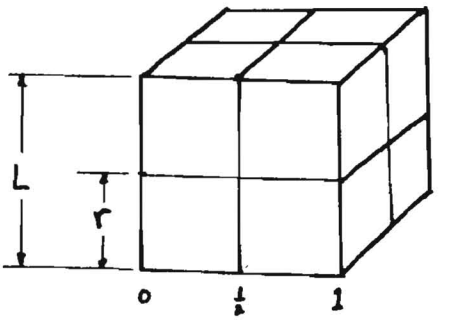


plane: (area)

$n = b^2 = 4$

$r = 1/b = 1/2$

$L = n\,r^2 = 1$ ft



space: (volume)

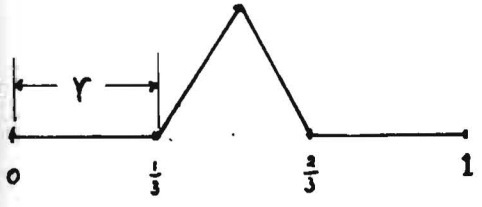
$n = b^3 = 8$

$r = 1/b = 1/2$

$L = n\,r^3 = 1$ ft

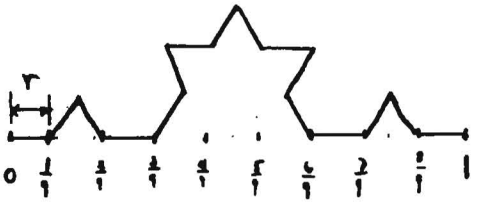
figure 13

In figure 14, the same process is used except that it is applied to a unit portion of the Koch curve. Note that the length of the unit curve turns out to be longer than one unit. In fact, a different length is produced for each stage of construction.



(N) r = 4(1/3) = 1.33

stage 2



(N) r = 16(1/9) = 1.778

stage 3

figure 14

For the limiting case we have $n = \infty$, $r \to 0$, and $L(n) = \infty$, which has been shown to be the case in the discussion on page 11. Since area and volume are zero for a curve, this method produces no new information for any other dimension greater than one.

How, then, can we show that a third of the Koch curve is shorter in length than the curve taken as a whole? One answer is to consider the Koch curve to be a shape which behaves as if it had a dimension somewhere between one and two.

SIMILARITY DIMENSION

An approach to dimension was formulated by Felix Hausdorff in 1919, and later A. S. Besicovitch put it into final form. The Hausdorff-Besicovitch dimension differs from standard dimension in that it may be a fraction. A general reference to the Hausdorff dimension is Ergodic and Information written by Patric Billingsley in 1965. The Hausdorff-Besicovitch dimension will serve to define fractal dimension; however, it is hard to handle rigorously. An understanding may be reached by considering the equation for unit measure: $N r^d = 1$, where $N = b^d$ subintervals of length $r = 1/b$, and d is the dimension of the unit shape. The equation is solved for d in terms of N and r:

$$N r^d = 1$$

$$r^d = 1/N$$

$$d \log(r) = \log(1/N)$$

$$d = - \log(N) / \log(r)$$

$$d = \log(N) / \log(1/r).$$

The latter form is called the similarity dimension and is often used to guess the Hausdorff dimension. The similarity dimension is less general than the Hausdorff dimension but will suffice for this paper. It is not a new idea but has received little attention in geometry courses because, for standard shapes, it reduces to the Euclidean dimension E; i.e., one based on intuition. For example, the dimension

of a line where N = 4 and r = 1/4 is given by d = log(n) / log(1/r) = log(4) / log(4) = 1 , the Euclidean dimension of a line. For a plane where N = 4 and r = 1/2, the dimension is given by d = log(4) / log(2) = 2, the Euclidean dimension of a plane. For the Koch curve, however, the result is a fraction. In the second stage, N = 4 and r = 1/3, then d = log(N) / log(1/r) = log(4) / log(3) ≈ 1.262.

Mandelbrot writes, that In general all d-dimensional parallelepipeds defined for $d \leq E$ (the Euclidean dimension) satisfy $r(N) = 1 / N^{1/d}$, thus $N r^{d} = 1$ or equvalently d = log(N) / log(1/r). For nonstandard shapes, such as the Koch curve, the only requirement needed to give the exponent of self-similarity formal meaning is that the shape be self-similar, i.e., that the whole may be split into N parts obtainable from it by a similarity ratio r (followed by displacement or by symmetry). The d obtained in this fashion always satisfies the inequality $0 \leq d \leq E$ [5, page 37].

Earlier it was shown how the equation for unit measure $N r^{d} = 1$ misbehaved for the Koch curve. It will now be applied with the newly found fractal dimension. A unit portion of the curve may be subdivided into N subintervals, each of length r. The unit measure in dimension d for any stage may now be found, as illustrated in figure 15, where d is equal to the actual value log(4)/log(3).

N = 4 ; r = 1/3

$4\,(1/3)^{d} = 1$

stage 1

N = 16 ; r = 1/9

$16\,(1/9)^{d} = 1$

stage 2

N = 64 ; r = 1/27

$64\,(1/27)^{d} = 1$

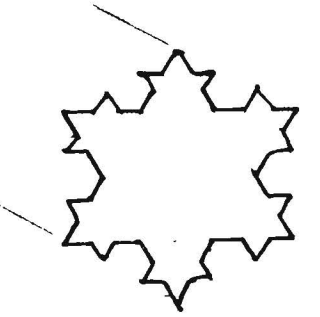
stage 3

figure 15

This is a reliable way to compare one portion of the curve to another. For example, in the third stage of construction of the triadic Koch curve, let a portion equal to 1/3 of the entire curve be of unit measure. The entire curve will then have a measure of three or three times as large as that portion (figure 16). Note that in the following figure, d has exact value log(4) / log(3).

N = 16 ; r = 1/9

$16\,(1/9)^{d} = 1$



N = 48 ; r = 1/9

$48\,(1/9)^{d} = 3$

figure 16

TOPOLOGICAL DIMENSION

The concepts of dimension discussed thus far have been metric notions. Topology defines dimension from a different point of view. In general, any set of objects is called a topological space if a collection of its subsets are singled out so that the collection has the following three properties: (1) The whole space and the empty set belong to the collection; (2) The union of any number of sets in the collection is also in the collection; (3) The intersection of any two sets in the collection is also in the collection. Topology extends the word neighborhood to include every open set. The effect of this generalization is to separate the notion of neighborhood from the idea of distance. Thus, a neighborhood is a collection of neighbors. From this point of view, the whole line may be viewed as a system of interlocking neighborhoods or open sets. The interlocking neighborhoods on the line determine what is called its topological structure. Two topological structures are considered to be the same if there is a one-to-one correspondence between them that preserves the interlocking system. Consider the example of a circle which is stretched like a rubber band into a distorted shape. Each point of the circle may take up a new position and distances between points may change, but the interlocking system of open sets remains unchanged [1, page 120].

Mandelbrot describes this aspect of topology with the following analogy: " ... all pots with two handles are of the same form because, if both are infinitely flexible and

compressible they can be molded into any other continuously without tearing any new opening or closing up any old one" [5, page 16].

A standard arc is a connected set that becomes disconnected if any single point is removed. A closed standard curve is a connected set that separates into standard arcs if two points are removed. The topological dimension Dt given to lines, arcs, and curves is one. For this reason the Koch curve is considered to have a topological dimension Dt = 1. The topology dimension is always an integer.

It was shown earlier that for standard shapes the similarity dimension d is also an integer equal to the Euclidean dimension. However, for nonstandard shapes, d need need not be an integer. The Koch curve, for example, has d ~ 1.2618 > Dt = 1. Historically, there has been no term to denote these sets whose d is greater than their topological dimension.

DEFINITION OF A FRACTAL

A fractal is by definition "a set for which the Hausdorff-Besicovitch dimension strictly exceeds the topological dimension" [5, page 15]. In most cases the Hausdorff-Besicovitch dimension is a fraction; however, it may be an integer. Mandelbrot coined "fractal dimension," denoted by D, rather than using "fractional dimension" to avoid this ambiguity. Thus, when dealing with fractal shapes, the fractal dimension D is considered to be synonymous with d, the similarity dimension.

Chapter 3

SELF-AVOIDING KOCH CURVES AND COASTLINES

The triadic Koch curve presented earlier is an example of a self-avoiding curve. Self-avoiding means that there is no overlapping or self-contact in any stage of the construction. This is important in defining the fractal dimension D because the similarity ratio requires that the whole be divided into disjoint parts. Coastlines are also self-avoiding and although coastlines are considerably more complex in their construction than the triadic Koch curve there are many similarities. The triadic Koch curve, is in many ways a rough model of a coastline.

An analogy describing maps of an island drawn to different scales was used in chapter I to illustrate why a standard jagged polygonal line represents a coastline of only a single scale. The standard jagged line will flatten out when it is magnified. However, the actual coastline when it is magnified or drawn to a larger scale reveals new detail. The jagged polygonal line has a single scale, whereas a coastline has for all practical purposes an infinite number of scales. The triadic Koch curve also has present an infinite number of scales. The triangle having side lengths equal to one, which is used as initiator for the triadic Koch curve, has a single scale. In the second stage, triangles having side lengths equal to 1/3 are piled on, and these triangles have a smaller scale. In the nth stage, triangles having side lengths equal to $(1/3)^{n-1}$ are piled on, and these triangles are of

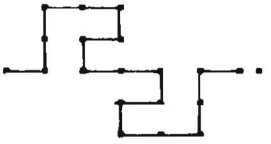
increasingly smaller scale. Thus the final curve has present an infinite number of scales below one.

Mandelbrot refers to a curve which is generated by replacing an initiator with an increasingly broken curve as a "teragon." The triadic Koch curve and a coastline are made up of irregular teragons of many different scales. The teragons which comprise the triadic Koch curve are very systematic in their irregularity, more so than a coastline. Thus, for this and other reasons, the triadic Koch curve is only a crude model of a coastline.

The general process used in creating the triadic Koch curve may be used to generate different families of Koch curves, and hence, of coastlines. By changing the initiator or generator, the teragons which determine a curve's form and fractal dimension may be changed. The initiator may be any carefully chosen M sided polygon. As the value of M increases, the resulting self-avoiding curves converge to a circle.

QUADRIC KOCH CURVES

Mandelbrot gives the term "quadric" to the family of Koch curves which uses a square for an initiator. Figure 17 shows the first five teragons of a quadric Koch curve with the generator  . Each of the eight line segments of the generator are scaled to 1/4 the length of the line they are replacing. In this case, the number of similar parts is N = 8, each part with a similarity ratio of r = 1/4 . The resulting fractal dimension is D = log(8)/log(4) = 1.5. The area contained within the curve is constant since the generator removes the same amount of area from the initiator as it adds to it.

Another example of a quadric Koch curve is shown in figure 18, but, it uses the generator  . This generator is such that N = 18 and the ratio r = 1/6. Thus, the fractal dimension D = log(18) / log(6) ~ 1.6131. The area of this curve also remains constant for the same reason cited above.

Notice that the teragons in figures 17 and 18 have the same basic shape. The similar overall outlines of these quadric Koch curves are due to a property called maximality and is based on the fact that the initiator is a square. Mandelbrot writes, " Consider all Koch generators that yield self-avoiding curves are traced on a square lattice made by straight lines parallel and perpendicular to [0,1], and in

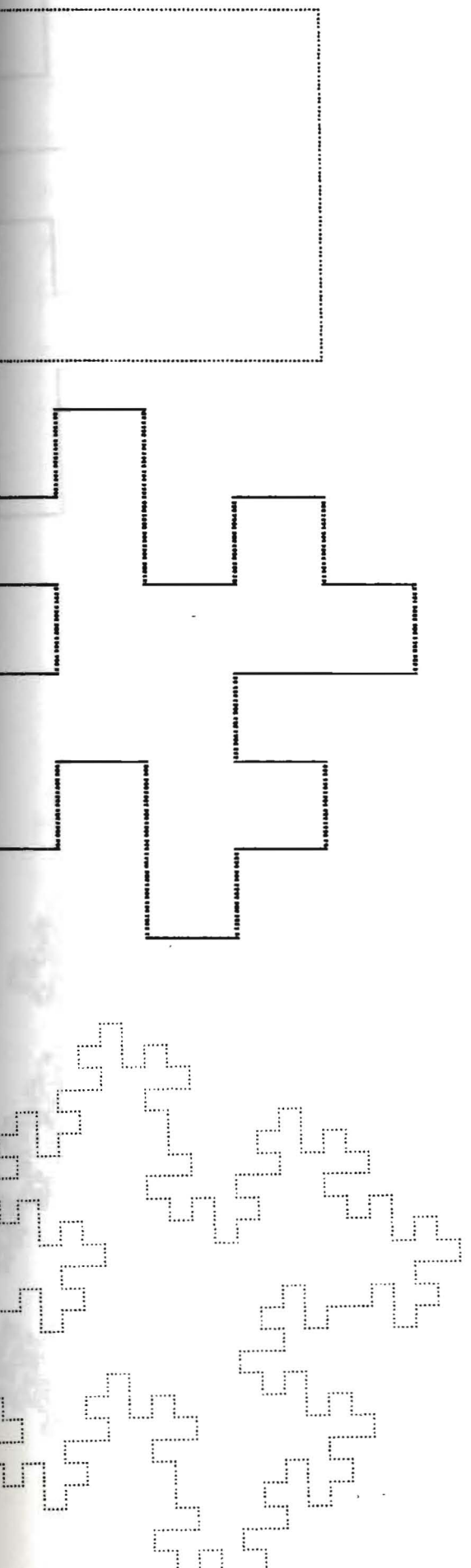
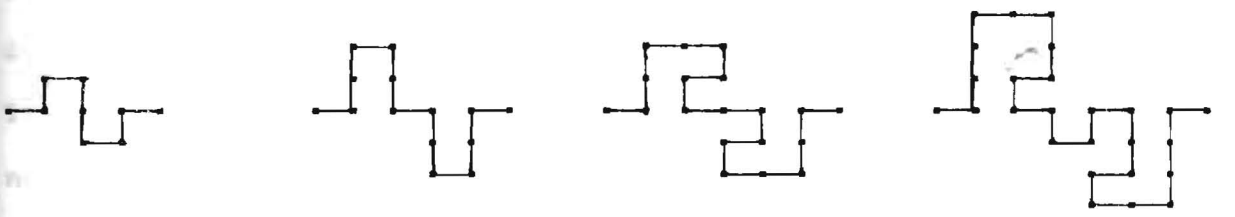


figure 17

figure 18

addition can be used with any initiator on the square lattice. We denote as maximal the generators that attain the highest possible value on N and hence of D. One finds that $N_{max} = b^2 / 2$ when b is even, while $N_{max} = (b + 1)^2 / 2$ when b is odd" [5, page 52]. Figure 19 illustrates a few of the many possible maximal generators.



b = 4 N = 8 | b = 5 N = 13 | b = 6 N = 18 | b = 7 N = 25

figure 19

Mandelbrot continues, "The maximal N and the number of alternative maximal polygons increase as the value of b increases. Generating self-avoiding maximal curves becomes increasingly difficult as D approaches 2 and is not possible for D = 2" [1, page 52]. Coastlines, however, are far from being maximal.

MORE ON COASTLINES

Another similarity which exists between Koch curves and coastlines is the problem encountered in trying to determine their length. Methods of measuring coastlines are dependent upon a yardstick approach. For example, consider a measuring device, such as a divider or a yardstick, set equal to a prescribed length e. The device is walked along the coastline, each new step starting where the previous step leaves off. The length of the coastline L(e) is equal to the number of steps multiplied by e , the length of the measuring device. A wheel having circumference e may also be used to measure the length of a coastline. The wheel is pushed along the coastline within a prescribed distance q which is proportional to e; i.e., the smaller the wheel the closer it may come to the coastline. The length L(e) is equal to the number of revolutions multiplied by the circumference e. The length obtained by these methods is only an approximation. When the "yardstick length" e of a measuring device is allowed to grow smaller and smaller, the coastlines "approximate length" L(e) will grow larger and larger without bound.

There are two cutoff scales between which the teragons are considered coastlines. Teragons of extreme scales which lie outside of this range are of different character. For example, the range of scales of interest to a physicist is much too small to be of use to a geographer or geomorphologist. Mandelbrot suggests that for coastlines, a

practical outer cutoff K might be the diameter of the smallest circle encompassing an island and the inner cutoff e might be 20 meters. In the case of the coast at Chelsea, Mandelbrot points out that an intermediate zone in which L(e) varies little and is of great practical use is from K = 20 meters down to e = 20 centimeters. Below 20 centimeters the measurements become affected by the irregularity of the stones. The dimension observed between the cutoff scales is an "effective dimension." For example, objects such as a veil, a thread, or a tiny ball are actually three dimensional. However, if they are small enough, physicists will often consider them to be "in effect" 2, 1, and 0 dimensional, respectively. Thus, a coastline may have different fractal dimensions depending upon the scale from which it is viewed.

Lewis Fry Richardson in 1961 conducted empirical studies relating to the variation of the length L(e) obtained by the divider or yardstick method and found that the polygonal "approximate length" of a coastline L(e) is roughly equal to $F\ e^{1-D}$ where the number of sides N is approximately $F\ e^{-D}$, each having length e. The approximate length L(e) varies with different values of e, whereas the measure F is independent of e. Since according to Richardson $N = F\ e^{-D}$, then $F = N\ e^{D}$, and it follows that $F\ e^{-D}\ e^{D} = F$. Thus, F, which Mandelbrot calls the "approximate measure in the dimension D," is independent of e. Actual data shows F to vary little with e.

For comparison, consider a unit portion of the triadic Koch

curve (figure 22) whose length and measure in dimension D can be determined exactly. The length $L(1/3) = N_1 e = 4/3$. The measure in dimension D is given by $N_D e^D = 4 (1/3)^D = 1$ where $D = \log(4) / \log(3)$. Richardson's equation then becomes $L(1/3) = 1 (1/3)^{1-D} = 4/3$, which is the value expected. For real coastlines, F cannot be determined exactly because coastlines are not self-similar in the strict sense.

Richardson noted that the exponent D may vary depending on the coastline chosen, and that D may vary depending on the portion of a coastline chosen, when the portion is considered separately. Richardson compared the lengths of common borders between countries by searching encyclopedias and noted that differences in the lengths reported for the common borders between Spain and Portugal and between the Netherlands and Belgium differed as much as 20%. The difference is in part due to different values of e. Richardson found that the graph of the approximate total length versus the yardstick length e for coastlines and for borders between countries did not stabilize, while for Euclidean shapes the method converged near a well-determined value. Coastline and borderline data, which Richardson plotted on double logarithmic paper, fell on a straight line of negative slope. The slope of this line may be used to estimate the value of 1 - D. The triadic Koch will serve again as an example. When the points corresponding to log(L(e)) and log(e) are plotted, one finds that they lie on a line having slope of approximately .262. For example, $L(1/3) = 16/3$, $L(1/9) = 64/9$, and $L(1/27) = 256/27$. The resulting

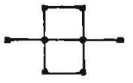
coordinates (log(r),log(L(r)) to three decimal places are (-.477,.727), (-.954,.852), and (-1.431,.977), respectively. Hence, the slope of the line which passes through these points is .262. Thus, 1 - D ~ .262 and D ~ 1.262 ~ log(4)/log(3) as expected. Richardson found that in general the expected value of D for actual coastlines is about 1.2.

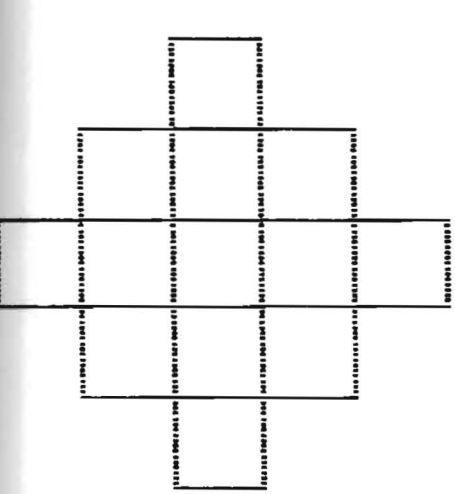
Richardson didn't consider the exponent D to have any special significance. However, Mandelbrot interprets Richardson's D as being a fractal dimension. He writes, "The Koch D is not an empirical but a mathematical constant. Therefore the argument for calling D a dimension becomes even more persuasive in the case of the Koch curve than in the case of coastlines" [1, page 36].

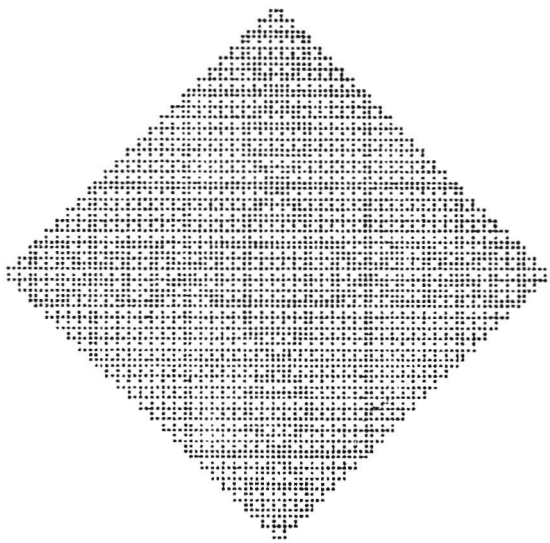
Peano Curves

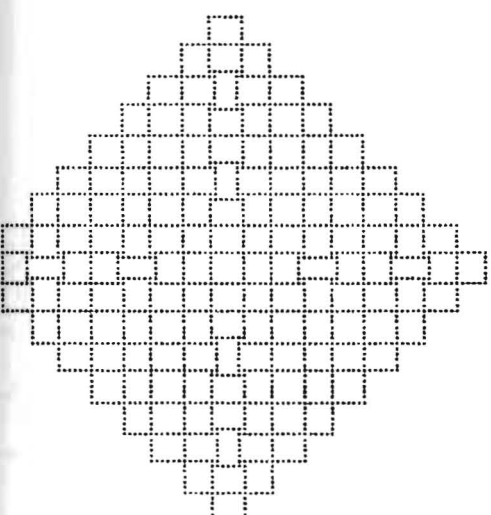
The Peano curve establishes a continuous correspondence between the straight line and the plane. A Peano curve may be considered to be a graphical representation of the mapping of an interval on to the whole of a square or a region of the plane. In this sense, a Peano curve is plane-filling.

It becomes increasingly difficult to construct self-avoiding curves with fractal dimensions approaching two. When D = 2, the curve must be viewed differently because 2 is also the Euclidean dimension of a plane. In fact, all classical definitions produce a dimension of 2 for Peano curves. The Peano curve is therefore a strange way of looking at a plane.

The concept was first written about by Peano in 1890. According to Mandelbrot, the first graphical implementation appeared in Moore in 1900. The curve is illustrated in figure 20. The initiator is the unit square. The generator  is made up of N = 9 parts, each having a similarity ratio of r = 1/3. Hence, the fractal dimension D = log(9) / log(3) = 2. For small values of n, i.e., the number of stages, the resulting curve is a checkerboard or grid pattern which becomes increasingly fine, covering more and more area as n approaches infinity. When the corners of the generator are rounded (figure 21), it is easier to see that the finite Peano curve is topologically the same dimension as a line, i.e., Dt = 1. Calling the curve a fractal is justified because 2 = D) Dt = 1.







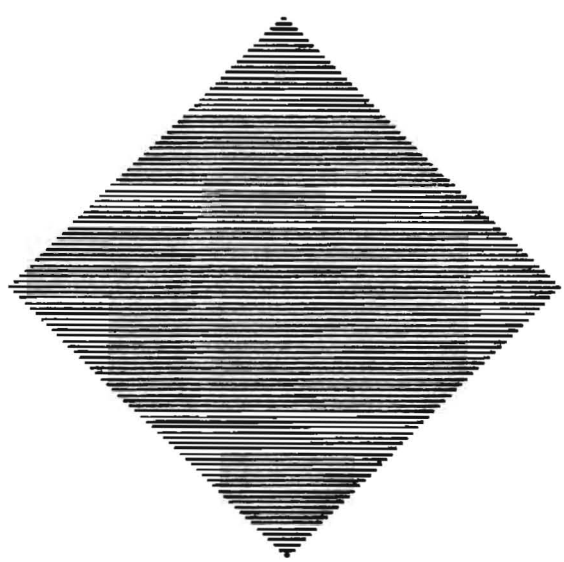
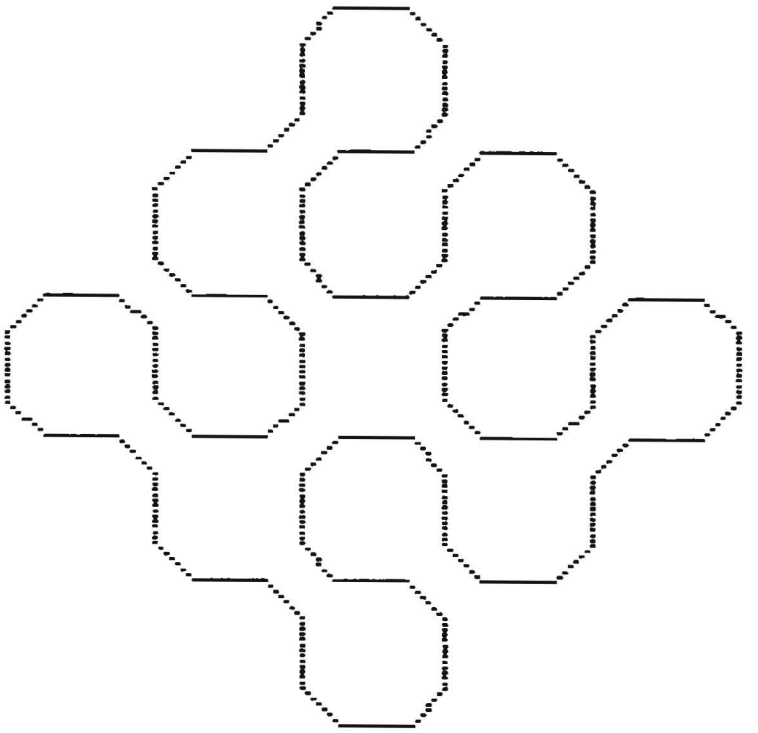


figure 20



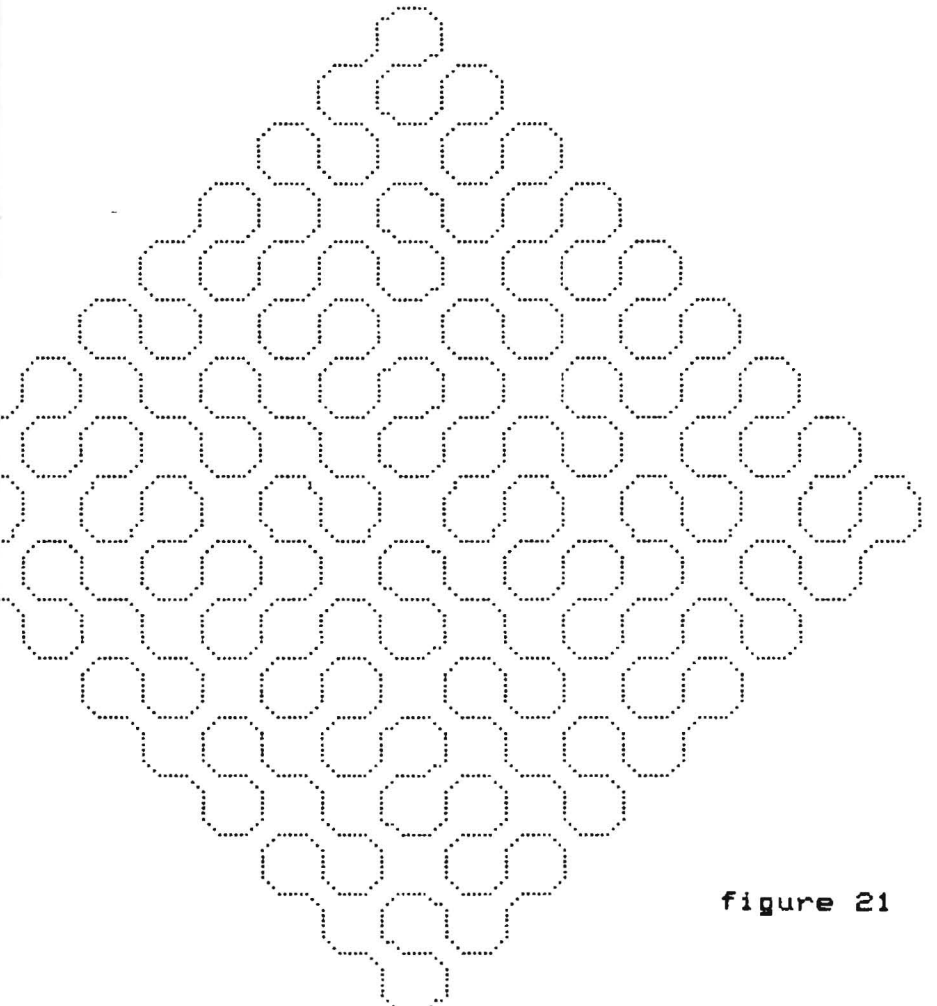


figure 21

In figure 20 the initiator is the unit square, and the distance between any of its parallel lines is one. In the second stage of construction the initiator is divided into 9 similar squares with four additional similar squares placed on the outside. The side length of each square in the grid is reduced from the whole by 1/3. Hence, the distance between parallel lines of the resulting grid is 1/3. In the third stage, the previous grid is divided into 117 similar squares with 20 additional squares placed on the outside. Each square is reduced from the squares of the previous stage by a ratio of 1/3 or reduced from the whole by a ratio of 1/9. The distance between parallel lines of the resulting grid, then, is 1/9. The distance between parallel lines of the grid produced in the nth stage is $1 / 3^{n}$. Thus, the distance between two parallel lines approaches zero as n approaches infinity. Self-contact then appears to be unavoidable. Hence, the limit Peano curve will always fill the plane or at least a portion thereof. The actual proofs, Mandelbrot points out, are delicate and central to the 1875-1925 crisis in mathematics and generated considerable controversy.

Historically, scientists and mathematicians considered the Peano curve to be pathological and "monstrous." Mandelbrot writes, "... Peano curves are far from being mathematical monsters with no concrete interpretation. If they fail to self-contact, they involve readily feasible and interpretable conjugate trees. These trees are good first order models of rivers, watersheds, botanical trees, and human vascular

systems" [1, page 68].

The human vascular system in order to supply blood to different parts of the body must pass within a small distance of every point of territory it serves. It makes a "plane sweeping motion" much like the Peano curve. Likewise, the river which completely drains a region is fed by a complex of tributaries which are fed by a complex of smaller tributaries which are fed by a complex of still smaller tributaries. The curve in figure 22 is a Peano curve which has been drawn to include each of the previous stages. It is somewhat reminiscent of a river with many different tributaries of many different scales.

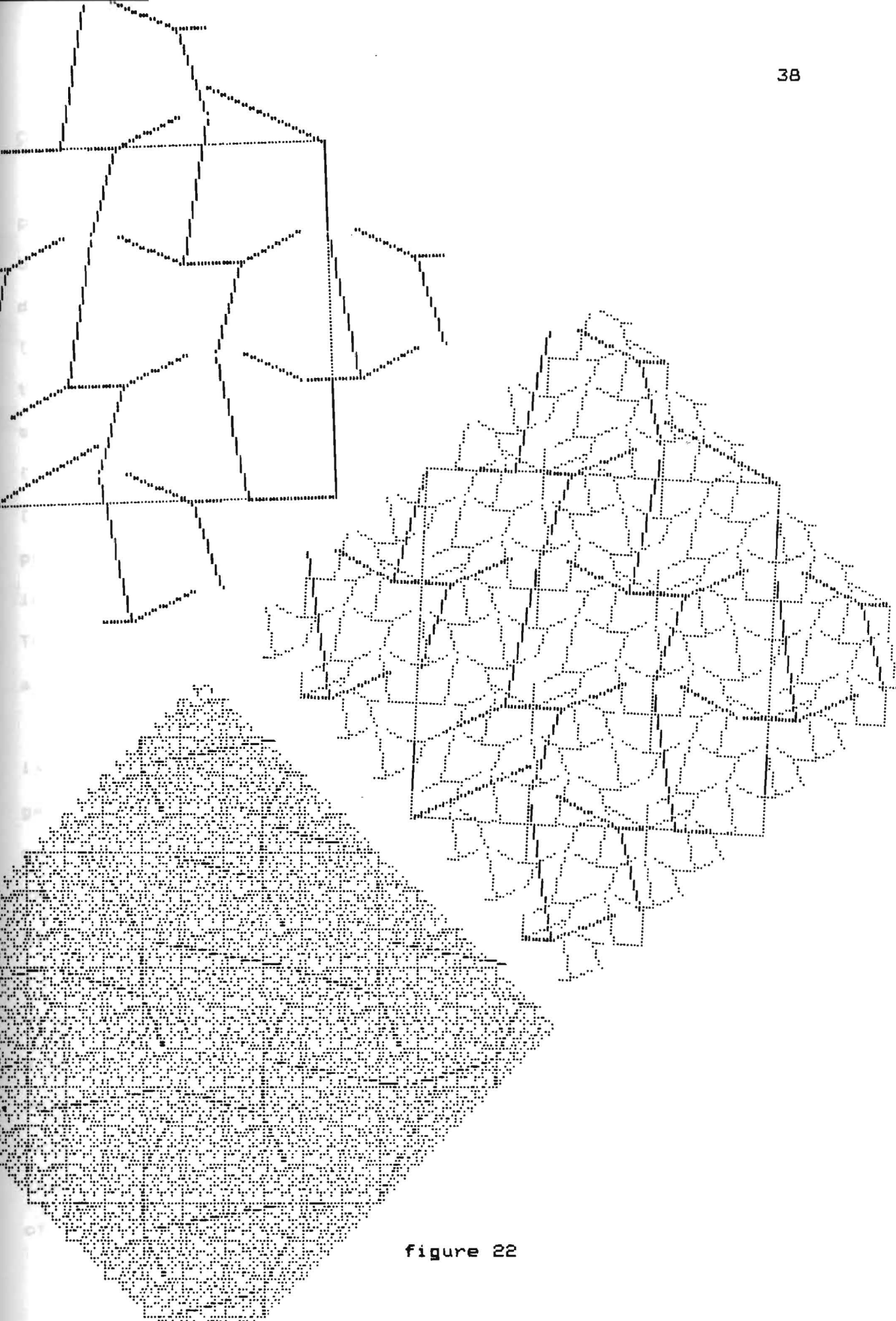


figure 22

CANTOR DUST

Cantor showed a one-to-one correspondence between the points of a line and the points of a plane by a process which begins with the closed interval [0,1]. The interval is divided into three subintervals: [0,1/3], (1/3,2/3), and [2/3,1]. Then, the middle open interval is removed, leaving the two subintervals [0,1/3] and [2/3,1]. Likewise, these subintervals are each divided into three more subintervals: [0,1/9], (1/9,2/9), [2/9,1/3], and [2/3,6/9], (6/9,7/9), [7/9,1]; again, the middle open intervals are removed. This process continues to infinity, dividing each closed interval into three subintervals and removing the middle open interval. The result, known as the Cantor set or Cantor discontinuum, is a discontinuous set of infinitely many closed intervals.

A representation of the first six stages of the Cantor set is illustrated in figure 23. The initiator [0,1] and the generator ——— ——— have been thickened into rectangles for clarity. The number of pieces is called the base b; in this case b = 3. The number of similar parts is N = 2 and the similarity ratio is r = 1/3. Hence, the fractal dimension D = log(2) / log(3) or approximately 0.6309. From a topological view, the Cantor set is of dimension Dt = 0 because it is not a continuum, or in the words of Mandelbrot, "...because any point is by definition cut from the other points without anything having to be removed to cut it. From this view point there is no difference between the Cantor set and finite sets of points" [1, page 78]. The Cantor set is a fractal because

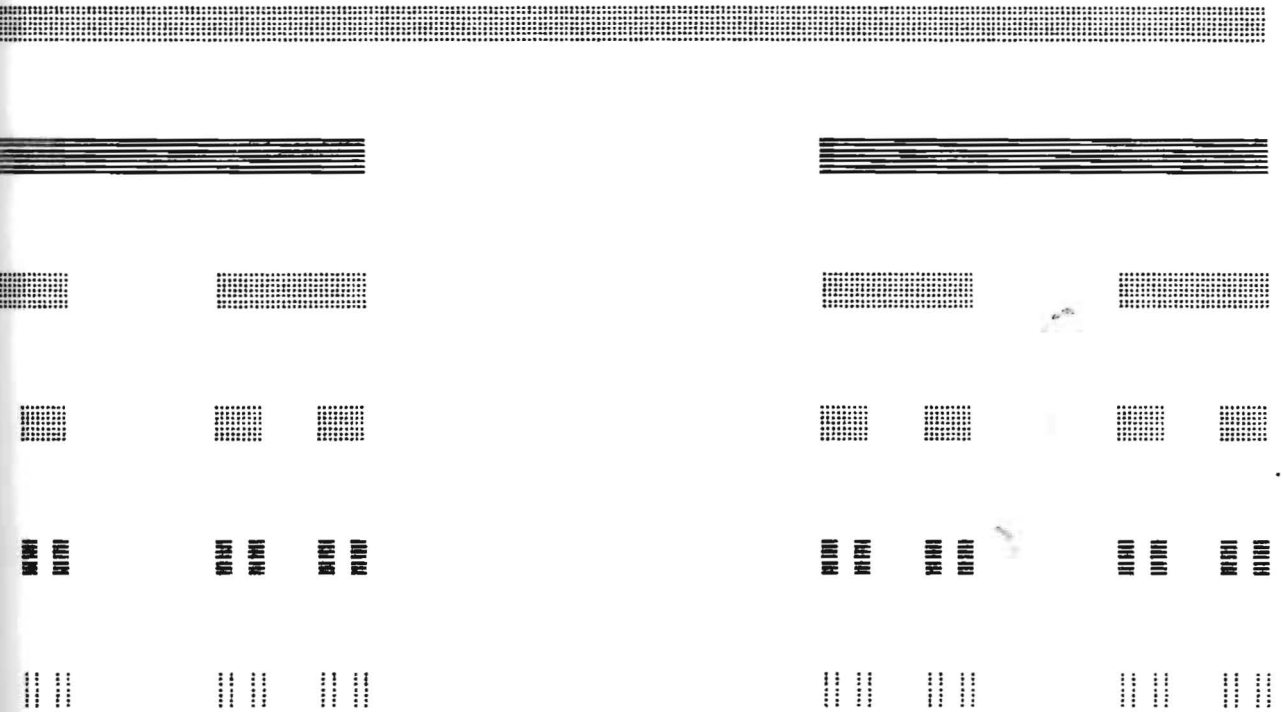


figure 23

.639 ~ D > Dt = 0. Furthermore, Mandelbrot terms these sets "Cantor fractal dusts" as a counterpart to "fractal curve" and "fractal surface."

"Curdling" is the term Mandelbrot gives to the process which generates the Cantor set. Consider a round bar of low density. The process is said to curdle the matter out of the middle third and into the outer thirds without changing their position. The outer thirds are called "precurds," and the

process leaves each precurd more dense than before. The process continues with the middle thirds of these bars curdling into their outer thirds, making four thinner precurds each having a greater density than they had previously. The curdling is repeated ad infinitum until there are infinitely many, infinitely thin, and infinitely dense precurds. The limit set is then called a "curd," and the space outside the curd is called "whey."

By using different rules for the curdling process, different fractal dimensions may be achieved such that $0 < D < 1$. For example consider a Cantor dust with the generator —— —— ——. In this case, $N = 3$, $r = 1/5$, and $D = \log(3) / \log(5) \sim .6826$. This generator, —— . . ——, is such that $N = 2$ and $r = 1/4$. Hence, the fractal dimension is $\log(2) / \log(4) = .5$.

SIGMA LOOPS

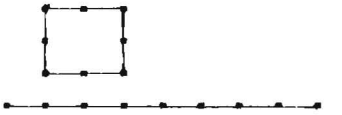
A large island is generally surrounded by smaller islands the number of which varies from a few to a multitude. The sizes of the surrounding islands are an important geographic characteristic. If large rock piles off shore are included in the count of surrounding islands along with individual rocks exposed above the surface or small scale patches of exposed surface which are isolated from the main shore by even the smallest amounts of water, then the total number of smaller islands which makes up the coastal form of the initial island tends to infinity.

The Koch process may be utilized to generate Koch fractal counterparts to the Cantor dust. For example, the generator

is made up of N = 16 similar parts each having a similarity ratio of r = 1/8. The fractal dimension for the curve (figure) is D = log(16) / log(8) = 4/3. The main generator is actually made up of two smaller generators: a coastline generator and an island generator. The coastline generator connects the points [0,1] and is made up of N' = 10 line segments. The island generator is the loop made up of N - N' = 6 line segments and "seeds" new islands. In order to generate fractals consistent with coastlines, the generator is chosen such that N' < N. The dimension of the island generator is D' = log(N') / log(1/r) which gives the coastline dimension of the individual islands. The dimension D = log(N) / log(1/r) is a measure of all the coastlines of all

the islands taken together. The cumulative coastline, not being connected, is not itself a curve but an infinite sum of loops. Hence, Mandelbrot proposes for it the term "sigma-loop."

FRACTAL DIMENSION AS A MEASURE OF FRAGMENTATION

The dimension D may measure fragmentation alone. The curve produced by the generator  (figure 24) has dimensions N = 16, N' = 8, and r = 1/8. The dimensions are D' = log(8) / log(8) = 1, and D = log(16) / log(8) = 1 1/3. In this case, the dimension D' is the same as the Euclidean dimension of line; thus, D' is not a measure of irregularity. Hence, D is a measure of fragmentation alone. When D' > 1, D' is a measure of irregularity, and D is a measure of both irregularity and fragmentation. Note that if the corners of the generator are rounded, then the curve will no longer be without tangents. Therefore, fractals are not always nonrectifiable.

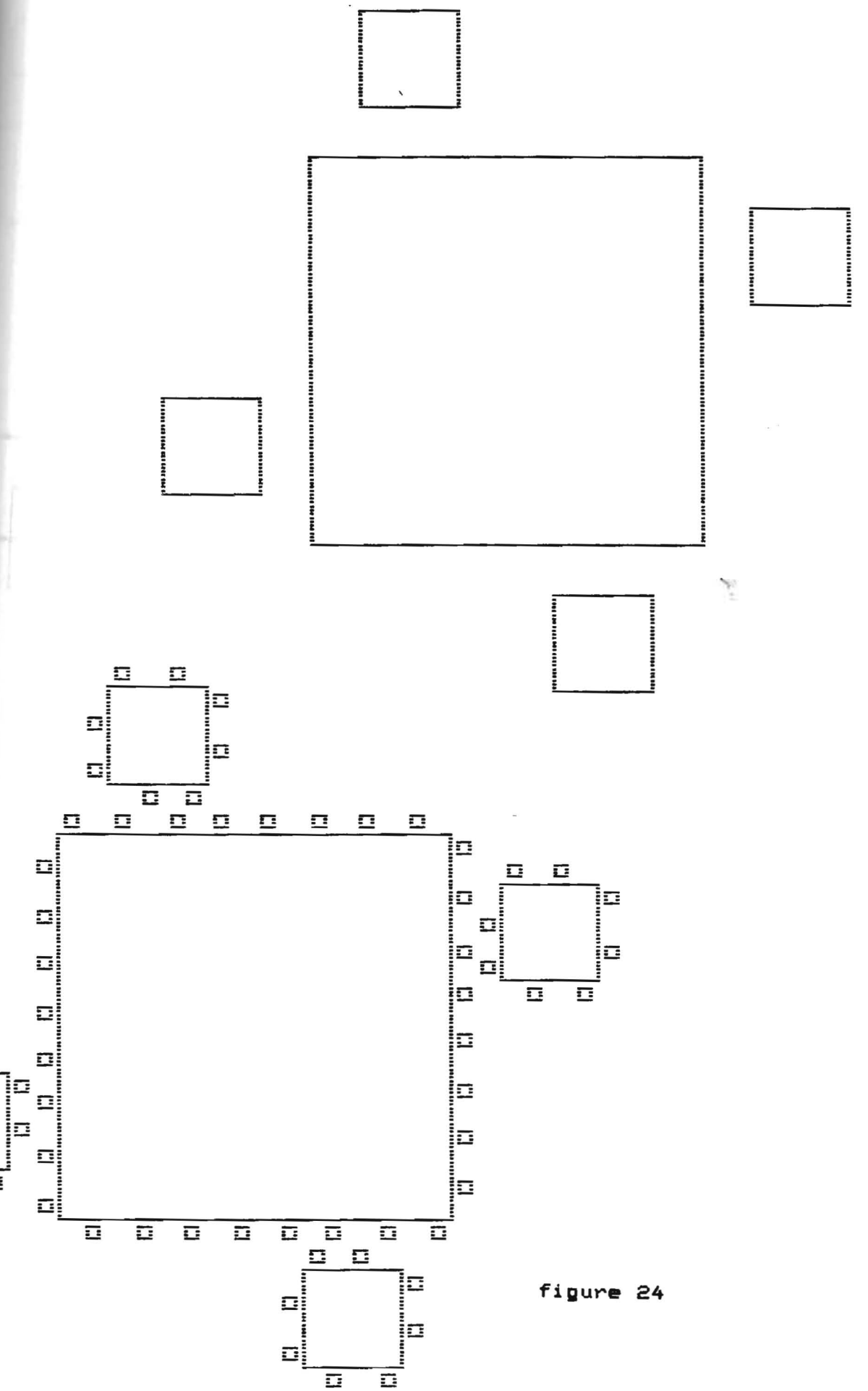


figure 24

CHANCE AND RANDOM FRACTALS

Figure 25 shows four stages of a quadric Koch curve with a poorly chosen generator; as a result, the curve overlaps.

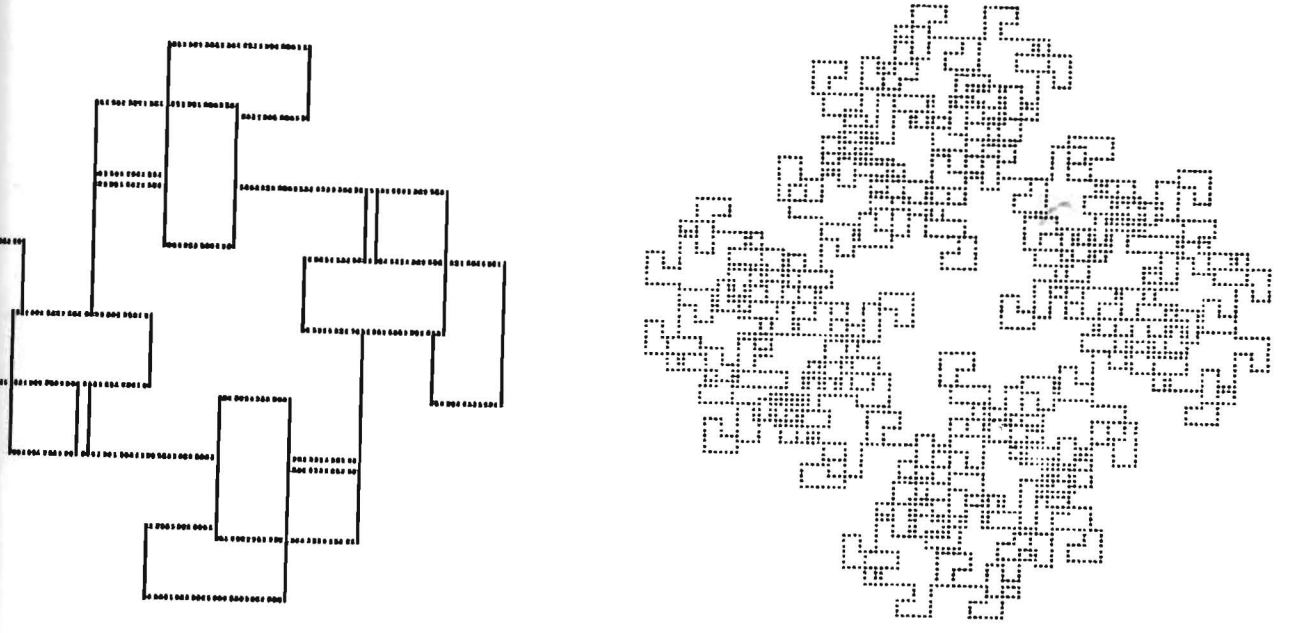


figure 25

The curve in figure 26 is due to a bug in the program written to generate the curve shown in figure 17. The curve is somewhat smoother and the overall pattern is less regular or structured. The generator and rules of placement are the same as in figure 17; however, as it turns out, the generator actually drawn on the printer or screen is  . The curve in figure 27 is the result of careless programming; the teragons are not self-avoiding. The curves presented in the

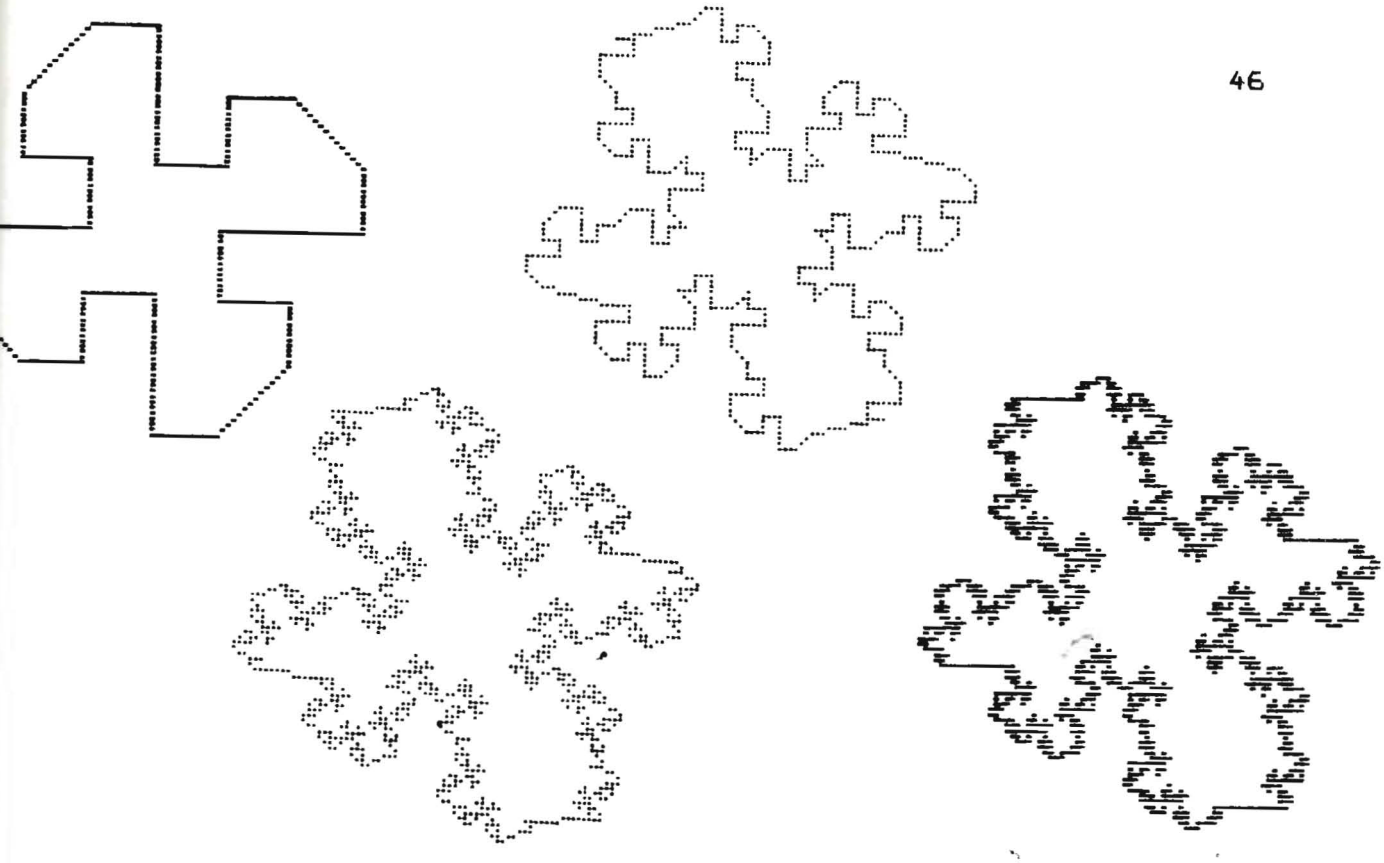


figure 26

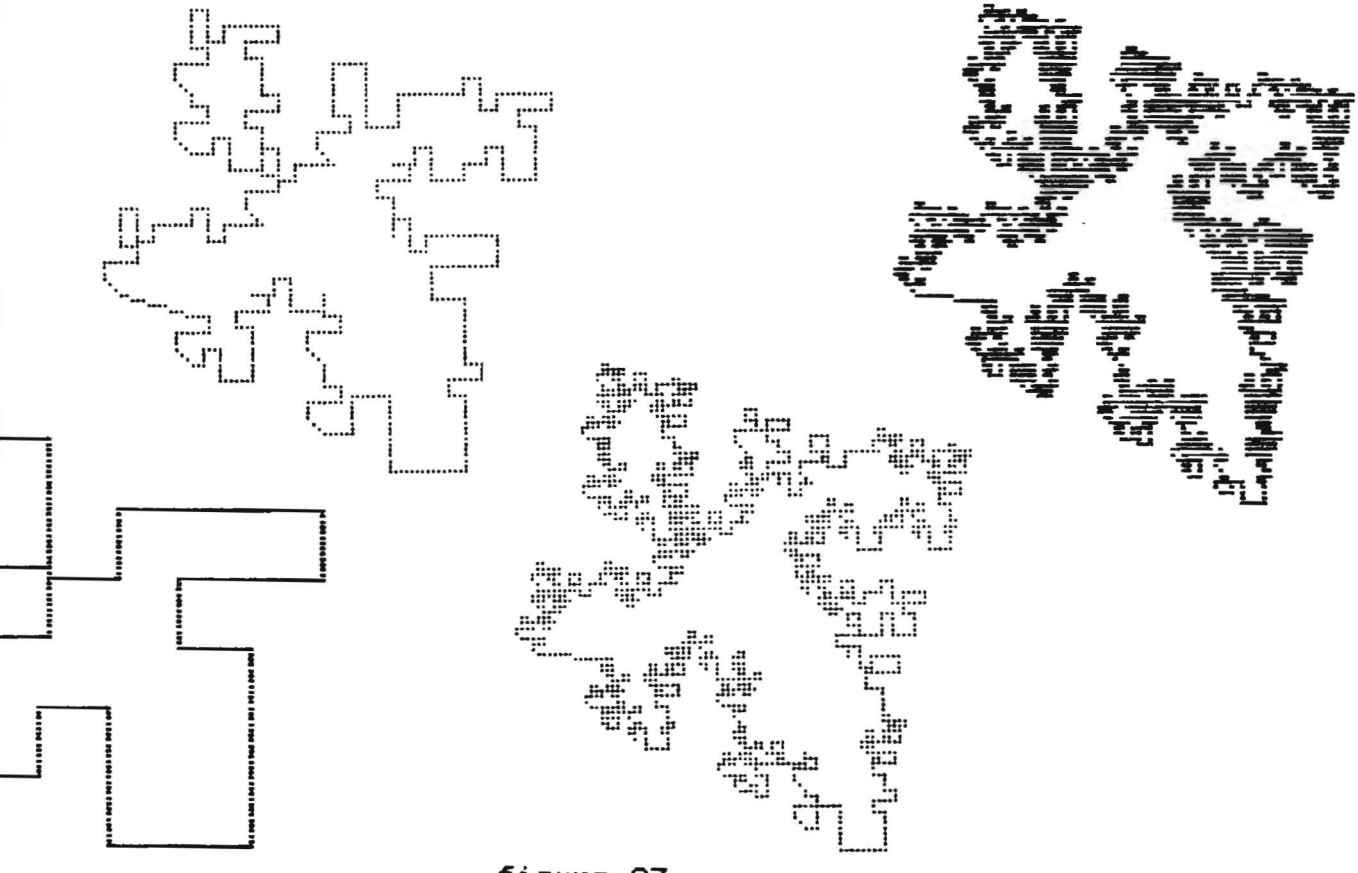


figure 27

last three figures have resulted merely by chance. If nothing else, these curves help to illustrate the power of the recursive method used to generate them, which Mandelbrot believes to be a clue to understanding many phenomena in nature.

Chance plays a vital role in the study of fractals, not the chance demonstrated by the curves mentioned above, but chance which has a statistical basis. Mandelbrot points out that "Although the basic fractal themes involve exclusively deterministic constructions, the full meaning and practical relevance of these themes are not apparent until one tackles random fractals" [5, page 200].

A stratified fractal is constructed by the superposition of layers, each involving finer detail. The easiest way to generate a random stratified fractal is to use different generators in different stages of the construction. For example, figure 28 illustrates a random Koch coastline where the generator used to construct the sigma loop in figure 24 is alternated at random with the generator used in figure 18. A model of a coastline may be improved by using a more complex deterministic algorithm. However, a curve generated in this manner is not a practical model because coastlines are molded through the years by many different forces all of which would be impossible to account for in a deterministic fashion. Hence, this method is of limited scope.

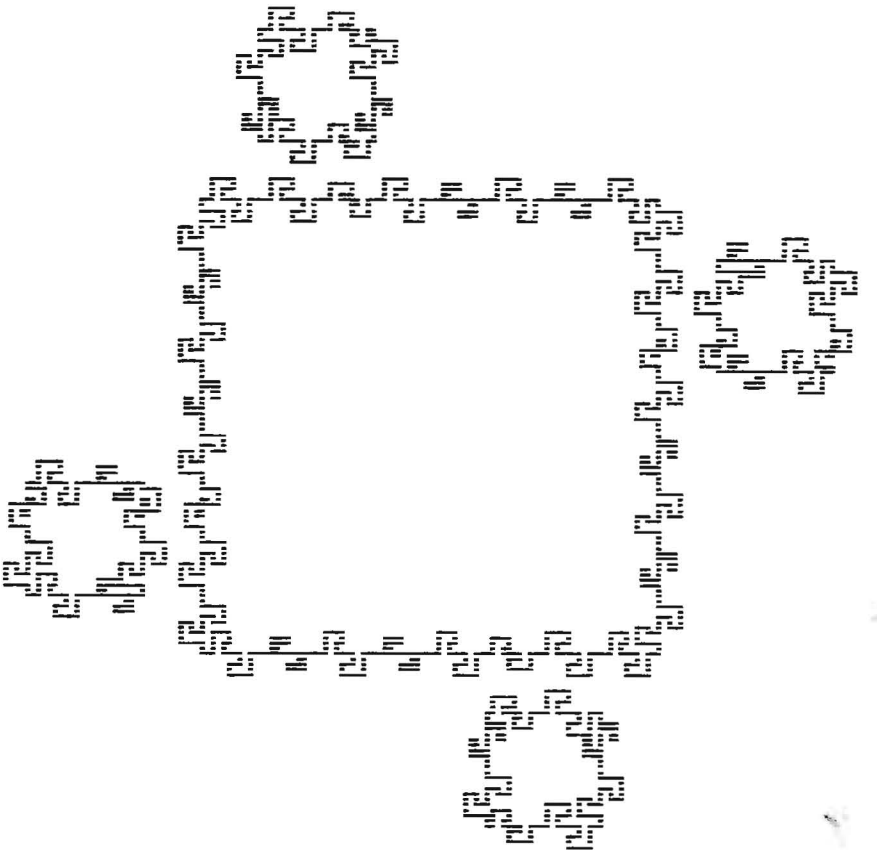


figure 28

The recursive methods used in this paper may be used with probability theory to invoke elements of chance. However, the rules that generate acceptable random curves are difficult to describe. Fractals which are generated by changing the shapes, sizes, and the order of a curve's parts, without having to inspect earlier loops, involve a nonconstrained form of chance. Constrained chance refers to a construction in which the later stages of a curve or coastline are constrained by the outcome of earlier stages. Mandelbrot and others have created fractal images which look like photographs of actual islands and mountains. Mandelbrot's mountains and coastlines involve probabilistic complications but are modeled in a first approximation by fractal surfaces ruled by Brownian chance.

Chapter 4

COMPUTER RECREATIONS

The triadic Koch curve was presented earlier in an algorithmic manner. The construction began with an initiator, each interval of which was replaced by a scaled generator. The points were renumbered and the process repeated again and again. This is an iterative method which may be implemented nicely in the programming language Pascal.

Pascal allows us to build a linked list of nodes dynamically; i.e., memory for storage is allocated during execution of a program. Thus, the programmer does not need to dimension an array to some maximum number beforehand. The nodes in this program are record variables defined to have three different fields: an x-field, a y-field, and a next-address field. The x and y fields are used to store the coordinates of a point. In Pascal the method of access to a node is through a pointer. A pointer may be thought of as the address of the portion in memory allocated to a node. LIST is a pointer set equal to the address of the first node. The next-address fields associated with each node are also pointers which point to the following node in the list. NIL is a special pointer which signifies the end of a list.

The program begins by generating a linked list of nodes corresponding to the three vertex points of the initiator and drawing the initial triangle. Then it begins the iterative process of calculating new points or generating new nodes which correspond to the scaled generator. Three such points

are generated for each pair of points already in the list. The three new nodes are then inserted into the list between the pair of nodes from which they were generated. With each iteration the resulting figure is drawn, adding more and more detail to the curve.

Points and nodes are declared as follows :

```
type  nodeptr = ^nodetype ;
      nodetype = record
          x := integer ;
          y := integer ;
          next := nodeptr
        end ;
var  p, list, k : nodeptr ; .
```

The preceding statements have defined p, LIST, and k to be pointers; they point to variables of type nodetype. Here is an example of how the different fields of a node may be accessed. A call to the standard function GETNODE, p := GETNODE, will allocate a new memory space with the above specifications and place the address of that location in p. Then, the x-field of this node may be set equal to 5 by the statement p^.x := 5. The next-address field is set equal to the value of the pointer q by the statement p^.next := q.

Another standard procedure, INSAFTER(x,y), will call GETNODE to create a new node, set the information fields equal to x and y, and then insert this new node into the list after the node pointed to by p. The function GETNODE and procedure

INSAFTER are presented below. A convenient reference covering this material is chapter five in Data Structures Using Pascal written by Aaron M. Tenenbaum and Moshe J. Augenstein in 1981.

```
function GETNODE : nodeptr ;

var p : nodeptr;

begin
  new(p); {creats new node and sets p equal to its address}
  getnode := p
end { function getnode } ;


procedure INSAFTER(p:nodeptr; X:integer; Y:integer) ;

var q:nodeptr ;

begin
   if p = nil
   then writeln('error void insertion')
   else begin
        p := getnode ;
        p^.X  := X ;
        q^.Y  := Y ;
        q^.next := p^.next ;
        p^.next := q
     end ;
 end { procedure insafter} ;
```

The main loop responsible for expanding the list is as follows :

```
      p := list ;

      while p^.next <> nil
         do begin
            k := p^.next ;
            findpts(p^.X,p^.Y,k^.X,k^.Y) ;
            p := k
          end {while..do begin} ;

       findpts(p^.X,p^.Y,list^.x,list^.y) ;.
```

The variable LIST in this program will always point to the first node in the list. Then, the first statement above sets the pointer p equal to the address of the first node in the list. The statement, while p^.next <> nil, checks for the end of the list, at which time control jumps out of the loop. The first time through the loop, the pointer k is set equal to the address of the second node in the list. The procedure FINDPTS accepts the coordinates stored in the first two nodes of the list and calculates the coordinates of three new points. Then, it calls INSAFTER which creates three new nodes and inserts them between the nodes pointed to by p and k. Next, the pointer p is set equal to the address of the second node and the process is repeated. The pointer k is set to the address of the third node; FINDPTS inserts three new points into the list, and so on, until the last node is detected. Finally, a call to FINDPTS outside the WHILE loop will insert three new points between the last and first nodes of the list. The schematic in figure 29 depicts a call to the procedure findpts given the coordinates of points 1 and 2. The coordinates of points 2, 3, and 4 are calculated and placed into nodes. The nodes are then inserted into the list.

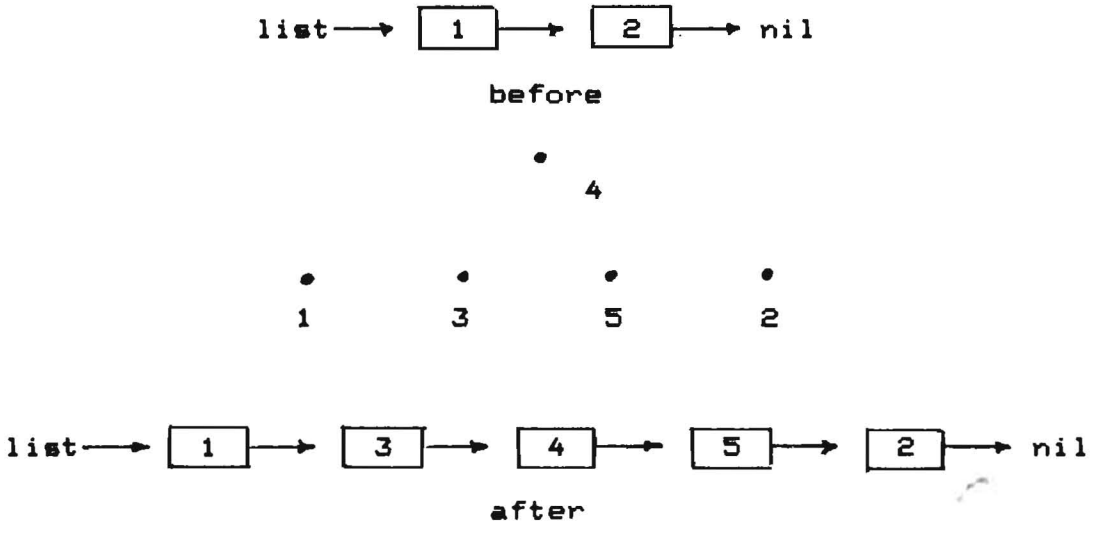


figure 29

The new points found by the procedure must be placed on the outside of the curve. The symmetry involved is such that the line segment between any pair of successive points, in any given stage of construction, may be placed into one of six different categories. For example, the line segment between two successive points, (x,y) and (kx,ky), may be designated as being horizontal to the right, horizontal to the left, up to the right, down to the right, up to the left, or down to the left; i.e., ER, EL, UR, DR, UL, DL, respectively. Figure 30 shows the line segment, its designation, and the condition to test for. Note the y coordinate axis is inverted when graphing on the computer. Because the screen is defined to have its origin at the upper left corner, the positive Y axis runs down from that point.

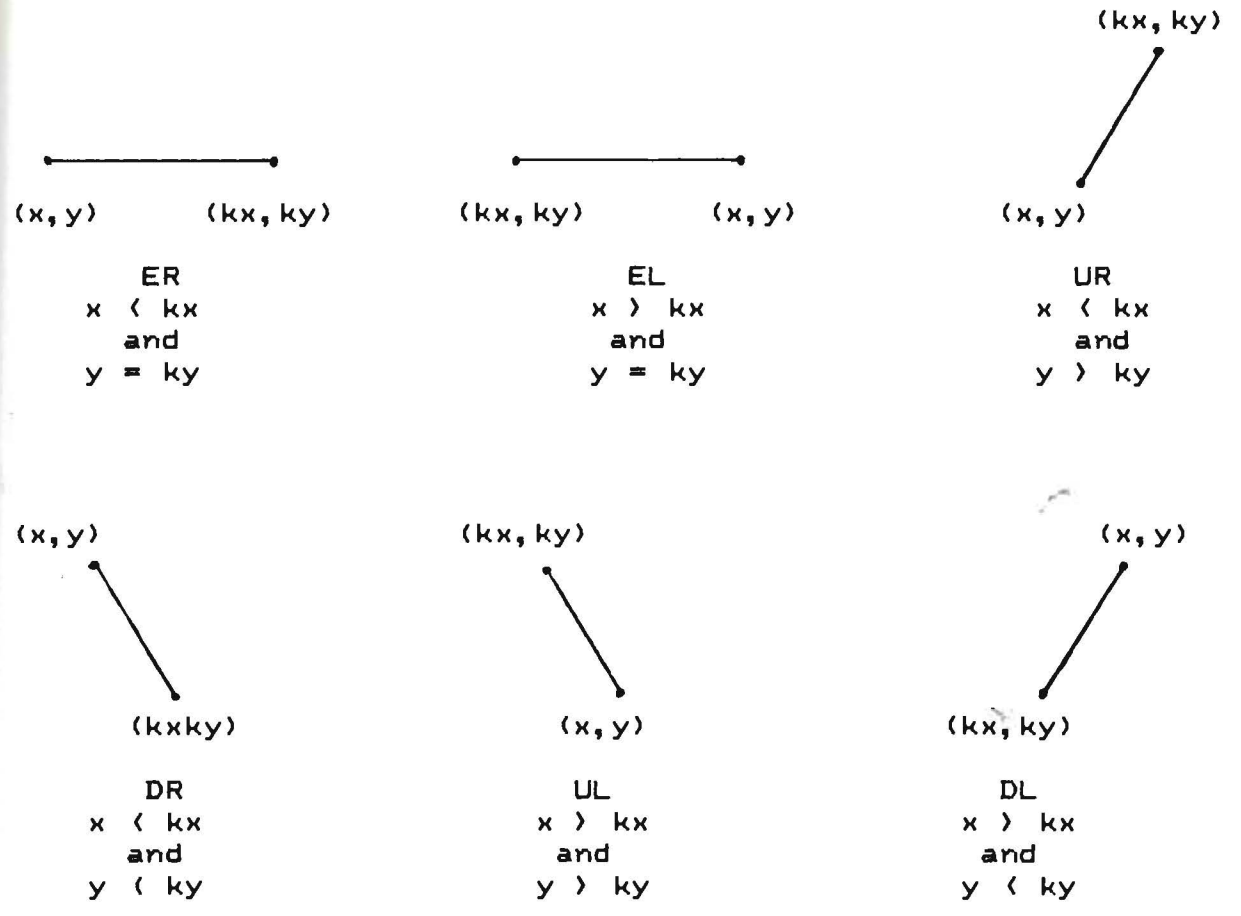


figure 30

The procedure FINDPTS uses two functions to determine the condition of a line. PLINE returns either the character "u", "d", or "e" depending on whether the line is up, down, or horizontal. The function DISTANCE returns either "l" or "r" for left or right. Consider the following example. Assume that the line segment between the points (x,y) and (kx,ky) is up and to the right. Then the points x1, y1, x2, y2, x3, and y3 may be calculated by the method in figure 31. The same general method is used for the five other cases involved. Recall that the y axis is inverted when in graph mode.

dx = | x − kx |

dy = | y − ky |

x1 = x + (1/3) dx

y1 = y − (1/3) dy

x3= x + (2/3) dx

y3 = y − (2/3) dy

x2 = x

y2 = y3

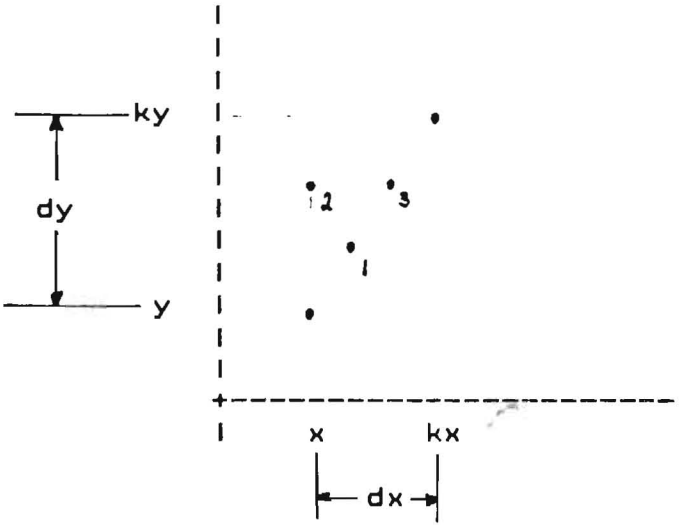


figure 31

The procedure responsible for drawing the curve is PDRAW. Given a linked list of points, PDRAW merely connects the points in an iterative manner, where the statement draw(x,y,kx,ky,color) draws a line from the point whose coordinates are (x,y) to the point whose coordinates are (kx,ky) in the color designated by an integer between 1 and 3. The program in its entirety is now presented.

```
program KOCH(input,output) ;

{  purpose - to draw consecutive stages of the triadic }
{            Koch curve.                               }

type nodeptr = ^nodetype ;
     nodetype = record
           x : integer ;
           y : integer ;
           next : nodeptr
        end ;


var  p,list,k        : nodeptr ;
```

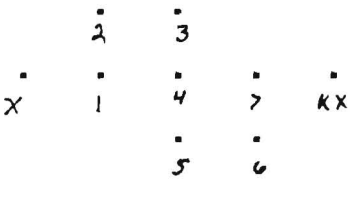
```
     i, x1,y1,x,y    : integer ;
         r           : real ;
       color         : integer ;
      linecolor      : integer ;
      backcolor      : integer ;
        palt         : integer ;
        stage        : char ;
        num          : integer ;


function GETNODE : nodeptr;
 var p : nodeptr ;
 begin
    new(p) ;
    getnode := p
  end { function getnode } ;


procedure INSAFTER(p:nodeptr; x:integer; y:integer) ;
 var q : nodeptr ;
 begin
 if p = nil
  then writeln('error void insertion')
  else begin
          q     :=  getnode ;
        q^.x    :=  x ;
        q^.y    :=  y ;
       q^.next  :=  p^.next ;
       p^.next  :=  q ;
     end;
  end { procedure insafter } ;


procedure PDRAW(color:integer) ;
 begin
   p := list ;
   while p^.next <> nil
      do begin
        K   :=  p^.next;
        draw(p^.x,p^.y,k^.x,k^.y,color) ;
        p   :=  p^.next ;
       end ;
   draw(p^.x,p^.y,list^.x,list^.y,color);
 end {procedure pdraw};



procedure FINDPTS(px:integer; py:integer; kkx:integer;
                                          kky:integer) ;

 var x,y,kx,ky,x1,y1,x2,y2,x3,y3 : integer ;
     dx,dy : real ;
     pl,di : char ;
```

```
function PLINE : char ;
begin
  if ky < y
    then pline := 'u'
    else if ky > y
           then pline := 'd'
           else pline := 'e'
   end { function pline };


function DIRECTION : char ;
begin
  if x < kx
    then direction := 'r'
    else direction := 'l'
 end { function direction };


begin    {   procedure  findpts  }

 x  := px ;
 y  := py ;
 kx := kkx ;
 ky := kky ;
 dx := abs(kx-x) ;
 dy := abs(ky-y) ;

 pl := pline ;
 di := direction ;

 if (pl = 'u') and (di = 'r')
   then begin
      x1 := x + round(0.3333 * dx) ;
      y1 := y - round(0.3333 * dy) ;
      x3 := x + round(0.6666 * dx) ;
      y3 := y - round(0.6666 * dy) ;
      x2 := x ;
      y2 := y3
     end
 else if (pl = 'u') and (di = 'l')
    then begin
     x1 := x - round(0.3333 * dx) ;
     y1 := y - round(0.3333 * dy) ;
     x3 := x - round(0.6666 * dx) ;
     y3 := y - round(0.6666 * dy) ;
     x2 := kx ;
     y2 := y1
   end

  else if (pl = 'd') and (di = 'r')
    then begin
     x1 := x + round(0.3333 * dx) ;
     y1 := y + round(0.3333 * dy) ;
```

```
      x3 := x + round(0.6666 * dx) ;
      y3 := y + round(0.6666 * dy) ;
      x2 := kx ;
      y2 := y1
    end

  else if (pl = 'd') and (di = 'l')
    then begin
      x1 := x - round(0.3333 * dx) ;
      y1 := y + round(0.3333 * dy) ;
      x3 := x - round(0.6666 * dx) ;
      y3 := y + round(0.6666 * dy) ;
      x2 := x ;
      y2 := y3
    end

  else if (pl = 'e') and (di = 'r')
    then begin
      x1 := x + round(0.3333 * dx) ;
      y1 := y ;
      x3 := x + round(0.6666 * dx) ;
      y3 := y ;
      x2 := round(x + dx/2) ;
      y2 := y - round(sqrt(dx*dx/9 - dx*dx/36)) ;
    end

     else begin
      x1 := x - round(0.3333 * dx) ;
      y1 := y ;
      x3 := x - round(0.6666 * dx) ;
      y3 := y ;
      x2 := x - round(dx/2) ;
      y2 := y + round(sqrt(dx*dx/9 - dx*dx/36)) ;
    end ;

  insafter(p,x3,y3) ;
  insafter(p,x2,y2) ;
  insafter(p,x1,y1) ;

end  { procedure findpts } ;
 begin              {  * * *  program   KOCH   * * *  }

  writeln('would you like each successive stage to be');
  writeln('drawn on the previous stage    Y or N ');
  readln(stage) ;
  writeln;
  writeln(' enter the X, Y coordinates for the lower left
                                                    vertex') ;
  readln(x,y) ;
  writeln;
  writeln('enter the distance between two of the initiators
                                                   vertexes') ;
  readln(r) ;
  writeln;
```

```
   writeln('how many stages ?  1-5);
   readln(num);
   clrscr;

 list  := getnode ;        { initialize linked list }
 list^.X := x ;
 list^.y := y ;
 list^.next := nil ;
 p := list ;

 x1 := round(x + r) ;      { puts the points of the initiator }
 y1 := y ;                 { into the linked list             }
 insafter(p,x1,y1);

x1 := round(x + r/2) ;
y1 := y - round( sqrt(r*r - r*r/4)) ;
insafter(p,x1,y1) ;
writeln('how many stages would you like') ;
readln(numstg) ;

backcolor := 0 ;
palt      := 3 ;
graphmode ;
graphbackground(backcolor) ;
color := 2 ;

pdraw(color) ;

for i := 1 to num
   do begin

        if color = 3
          then color := 0;
        color := color + 1 ;

        p := list ;

        while p^.next <> nil
           do begin
                k := p^.next ;
                findpts(p^.x,p^.y,k^.x,k^.y) ;
                p := k ;
              end ;
         findpts(p^.x,p^.y,list^.x,list^.y) ;

        readln ;

        if stage = 'N'
          then graphmode ;
        pdraw(color) ;

      end ;

 end  { program Koch } .
```

The programs which generate the quadric Koch curves in figures 17 and 18 are similar to the preceding program. The only changes made were to the code responsible for the initiator and to the procedure FINDPTS. The initiator is changed from a triangle to a square. The procedure FINDPTS uses the following numbering system corresponding to the generator:



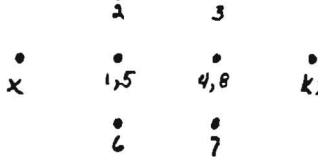
The procedure for the curve in figure 17 is as follows:

```
procedure FINDPTS(px:integer; py:integer; kkx:integer;
                                          kky:integer; alt:char) ;

   var  x,y,x1,y1,x2,y2,x3,y3,kx,ky : integer ;
        x4,y4,x5,y5,x6,y6,x7,y7 :integer ;
        dx,dy : real ;
        pl,di : char ;

   function PLINE : char ;
     begin
        If ky < y
        then pline := 'u'
        else if ky > y
            then pline := 'd'
            else pline := 'e'
      end  { function pline} ;

  function direction : char ;
    begin
        if x < kx
        then direction := 'r'
        else direction := 'l'
    end { function direction } ;

begin

  X := px ;
  Y := pY ;
  kx := kkx ;
  ky := kky ;
```

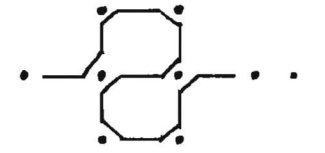
```
dx := abs(kx-x) ;
dy := abs(ky-y) ;

pl := PLINE ;
di := direction ;

   if pl = 'u'
     then begin
          x1 := x ;
          y1 := y - round(0.25 * dy)  ;
          x2 := x - round(0.25 * dy)  ;
          y2 := y1 ;
          x3 := x2 ;
          y3 := y - round(0.5 * dy)  ;
          x4 := x ;
          y4 := y3 ;
          x5 := x + round(0.25 * dy)  ;
          y5 := y4 ;
          x6 := x5 ;
          y6 := y - round(0.75 * dy)  ;
          x7 := x ;
          y7 := y6 ;
        end

   else if pl = 'd'
       then begin
          x1 := x ;
          y1 := y + round(0.25 * dy)  ;
          x2 := x + round(0.25 * dy)  ;
          y2 := y1 ;
          x3 := x2 ;
          y3 := y + round(0.5 * dy)  ;
          x4 := x ;
          y4 := y3 ;
          x5 := x - round(0.25 * dy)  ;
          y5 := y4 ;
          x6 := x5 ;
          y6 := y + round(0.75 * dy)  ;
          x7 := x ;
          y7 := y6 ;
        end

  else if (pl = 'e') and (di = 'r')
      then begin
          x1 := x + round(0.25 * dx)  ;
          y1 := y ;
          x2 := x1 ;
          y2 := y - round(0.25 * dx)   ;
          x3 := x + round(0.5 * dx)  ;
          y3 := y2 ;
          x4 := x3 ;
          y4 := y ;
          x5 := x4 ;
          y5 := y + round(0.25 * dx)  ;
```

```
          x6 := x + round(0.75 * dx)   ;
          y6 := y5 ;
          x7 := x6 ;
          y7 := y  ;
        end
    else if (pl = 'e') and (di = 'l')
      then begin
          x1 := x - round(0.25 * dx)   ;
          y1 := y ;
          x2 := x1 ;
          y2 := y + round(0.25 * dx)    ;
          x3 := x - round(0.5 * dx)    ;
          y3 := y2 ;
          x4 := x3 ;
          y4 := y ;
          x5 := x4 ;
          y5 := y - round(0.25 * dx)   ;
          x6 := x - round(0.75 * dx)   ;
          y6 := y5 ;
          x7 := x6 ;
          y7 := y ;
        end ;

    INSAFTER(p,x7,Y7)  ;
    INSAFTER(p,x6,Y6)  ;
    INSAFTER(p,x5,Y5)  ;
    INSAFTER(p,x4,Y4)  ;
    INSAFTER(p,x3,Y3)  ;
    INSAFTER(p,x2,Y2)  ;

    if alt = 'y'
      then INSAFTER(p,x,Y)
      else INSAFTER(p,x1,Y1) ;

end { function findpts };
```

To produce the program for the Peano curve (figure 20) only changes to the procedure FINDPTS are needed. The numbering of the generator is such that two points are numbered twice:



The Peano curve with rounded corners (figure 21) uses the same generator and numbering scheme as above. Hence, the procedure

FINDPTS for the two programs are the same; however, modification of the procedure PDRAW is required. The following example shows the figure which is actually drawn on the screen. The generator is shown as black dots and the result of PDRAW is shown as a curve: 

The procedure FINDPTS and PDRAW are as follows:

```
procedure FINDPTS(px:integer; py:integer; kkx:integer; kky:
                                                    integer) ;

var  x,y,x1,y1,x2,y2,x3,y3,kx,ky : integer ;
   x4,y4,x5,y5,x6,y6,x7,y7,x8,y8 :integer ;
   dx,dy : real ;
   pl,di : char ;

function PLINE : char ;
 begin
   If ky < y
     then pline := 'u'
     else if ky > y
            then pline := 'd'
            else pline := 'e'
   end  { function pline} ;

function direction : char ;
 begin
   if x < kx
     then direction := 'r'
     else direction := 'l'
 end { function direction } ;

begin  { function findpts }

  X := px ;
  Y := pY ;
  kx := kkx ;
  ky := kky ;
  dx := abs(kx-x) ;
  dy := abs(ky-y) ;

  pl := PLINE ;
  di := direction ;
```

```
  if pl = 'u'
    then begin
        x1 := x ;
        y1 := y - round(0.3333 * dy) ;
        x2 := x - round(0.3333 * dy)  ;
        y2 := y1 ;
        x3 := x2 ;
        y3 := y - round(0.6666 * dy) ;
        x4 := x ;
        y4 := y3 ;
        x5 := x1 ;
        y5 := y1 ;
        x6 := x + round(0.3333 * dy) ;
        y6 := y1 ;
        x7 := x6 ;
        y7 := y3 ;
        x8 := x4 ;
        y8 := y4 ;
      end

  else if pl = 'd'
      then begin
        x1 := x ;
        y1 := y + round(0.3333 * dy) ;
        x2 := x - round(0.3333 * dy)  ;
        y2 := y1 ;
        x3 := x2 ;
        y3 := y + round(0.6666 * dy) ;
        x4 := x ;
        y4 := y3 ;
        x5 := x1 ;
        y5 := y1 ;
        x6 := x + round(0.3333 * dy) ;
        y6 := y1 ;
        x7 := x6 ;
        y7 := y3 ;
        x8 := x4 ;
        y8 := y4 ;
      end
else if (pl = 'e') and (di = 'r')
      then begin
        x1 := x + round(0.3333 * dx) ;
        y1 := y ;
        x2 := x1 ;
        y2 := y - round(0.3333 * dx) ;
        x3 := x + round(0.6666 * dx) ;
        y3 := y2 ;
        x4 := x3 ;
        y4 := y ;
        x5 := x1 ;
        y5 := y1 ;
        x6 := x1 ;
        y6 := y + round(0.3333 * dx) ;
        x7 := x4 ;
```

```
            y7 := y6 ;
            x8 := x4 ;
            y8 := y4 ;
           end

      else if (pl = 'e') and (di = 'l')
        then begin
            x1 := x - round(0.3333 * dx) ;
            y1 := y ;
            x2 := x1 ;
            y2 := y - round(0.3333 * dx) ;
            x3 := x - round(0.6666 * dx) ;
            y3 := y2 ;
            x4 := x3 ;
            y4 := y ;
            x5 := x1 ;
            y5 := y1 ;
            x6 := x1 ;
            y6 := y + round(0.3333 * dx) ;
            x7 := x4 ;
            y7 := y6 ;
            x8 := x4 ;
            y8 := y4 ;
          end ;

      INSAFTER(p,x8,Y8) ;
      INSAFTER(p,x7,Y7) ;
      INSAFTER(p,x6,Y6) ;
      INSAFTER(p,x5,Y5) ;
      INSAFTER(p,x4,Y4) ;
      INSAFTER(p,x3,Y3) ;
      INSAFTER(p,x2,Y2) ;
      INSAFTER(p,x1,Y1) ;

end { function findpts };
procedure PDRAW(color:integer) ;
var    i,x,y,kx,ky,dx,dy : integer ;
       pl,di      : char ;

function PLINE : char ;
 begin
   If ky < p^.y
     then pline := 'u'
     else if ky > p^.y
            then pline := 'd'
            else pline := 'e'
   end  { function pline} ;

function direction : char ;
 begin
   if p^.x < kx
     then direction := 'r'
     else direction := 'l'
 end { function direction } ;
```

```
begin    { procedure pdraw }

   p := list ;
   X := p^.x ;
   Y := p^.Y ;
   kx := p^.next^.x ;
   ky := p^.next^.y ;
   dx := abs(kx-x) ;
   dy := abs(ky-y) ;
   if dx = 0
     then dx := dy
     else dy := dx ;

while p^.next <> nil
  do begin

    pl := PLINE ;
    di := direction ;
    if (pl = 'u')
           then begin
              draw(x,y,kx,ky+round(dy*0.25),color);
              x := kx ;
              y := ky + round(dy*0.25)
             end;
           if (pl = 'd')
             then begin
                draw(x,y,kx,ky-round(dy*0.25),color) ;
                x := kx ;
                y := ky - round(dy*0.25)
              end;

           if (pl = 'e') and (di = 'r')
             then begin
                draw(x,y,kx-round(dx*0.25),ky,color);
                x := kx - round(dx*0.25) ;
                y := ky
                end ;
           if (pl = 'e') and (di = 'l')
                 then begin
                    draw(x,y,kx+round(dx*0.25),ky,color) ;
                    x := kx + round(dx*0.25) ;
                    y := ky
                  end ;

    for i := 1 to 14
      do if p^.next <> nil
         then begin

         if (i/2 <> round(i/2))
          then begin
            p := p^.next ;
            kx := p^.next^.x ;
            ky := p^.next^.y
           end ;
```

```
      pl := PLINE ;
      di := direction ;

       if (pl = 'u')
         then begin
           draw(x,y-round(dy*0.25),kx,ky+round(dy*0.25)
                                                  ,color);
           x := kx ;
           y := ky + round(dy*0.25);
          end

          else if (pl = 'd')
             then begin
                draw(x,y+round(dy*0.25),kx,ky-round(dy
                                          *0.25),color);
                x := kx ;
                y := ky - round(dy*0.25)
              end

              else if (pl = 'e') and (di = 'r')
                then begin
                  draw(x,y,kx-round(dx*0.25),ky,color);
                   x := kx - round(dx*0.25) ;
                   y := ky
                end
                else begin
                  draw(x,y,kx-round(dx*0.25),ky,color);
                   x := kx - round(dx*0.25) ;
                   y := ky
                end ;
 end ;

  p := p^.next ;
  kx := p^.next^.x ;
  ky := p^.next^.y ;

if p^.next <> nil
 then begin

  if (pl = 'u')
   then begin
      draw(x,y-round(dy*0.25),kx,ky,color) ;
      x := kx ;
      y := ky
    end;

    if (pl = 'd')
      then begin
         draw(x,y+round(dy*0.25),kx,ky,color) ;
         x := kx ;
         y := ky
       end;

    if (pl = 'e') and (di = 'r')
```

```
            then begin
               draw(x+round(dx*0.25),y,kx,ky,color) ;
               x := kx ;
               y := ky
            end;

           if (pl = 'e') and (di = 'l')
             then begin
               draw(x-round(dx*0.25),y,kx,ky,color) ;
               x := kx ;
               y := ky
             end;

            p := p^.next ;
            kx := p^.next^.x ;
            ky := p^.next^.y ;
          end ; {if then begin }
       end;
    draw(x,y,list^.x,list^.y,color);

end  {procedure PDRAW};
```

BIBLIOGRAPHY

[1] Adler, Irving. The New Mathematics. New York: The John Day Company, 1958.

[2] Benson, V. Russell. Euclidean Geometry and Convexity. New York: McGraw-Hill Book Company, 1966.

[3] Davis, M. "Interview: Benoit B. Mandelbrot." Omni, Feb. 1984, pp. 64-66.

[4] Hurewicz, Witold and Henry Wallman. Dimension Theory. Princeton, New Jersey: Princeton Univ. Press, 1948.

[5] Mandelbrot, Benoit B. The Fractal Geometry of Nature. New York: W.H.Freeman and Company, 1982.

[6] McDermott, Jeanne. "Geometrical Forms Known as Fractals Find Sense in Chaos" Smithsonian, Dec. 1983, pp. 110-117.

[7] Munem, A. Mustafa and David J. Foulis. Calculus with Analytic Geometry. New York: Worth Publishers Inc., 1978.