Generation and Display of Geometric Fractals in 3-D

Alan Norton
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

We present some straightforward algorithms for the generation and display in 3-D of fractal shapes. These techniques are very general and particularly adapted to shapes which are much more costly to generate than to display, such as those fractal surfaces defined by iteration of algebraic transformations. In order to deal with the large space and time requirements of calculating these shapes, we introduce a boundary-tracking algorithm particularly adapted for array-processor implementation. The resulting surfaces are then shaded and displayed using z-buffer type algorithms. A new class of displayable geometric objects, with great diversity of form and texture, is introduced by these techniques.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Curve, surface, solid, and object representations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Color, shading, shadowing, and texture.

General Terms: Algorithms, theory.

Additional Key Words and Phrases: fractal dimension, surface determination, iteration, invariant surface, quaternions.

Introduction

Until recently the geometric shapes of interest in engineering, science, and mathematics generally were constructed from the simple, smooth objects of classical geometry. However, a new approach to geometry has now arisen, largely through the efforts of its originator, B. Mandelbrot. He singled out a class of shapes, named them fractals, and developed nonrandom and random fractal models to simulate (and in some cases, to explain) the roughness and fragmentation of diverse aspects of nature. His fractal models of terrain prove of wide application in computer graphics [2,6,8].

Of special interest are the nonrandom fractals which are obtained by the iteration of algebraic functions. The mathematics of iteration has a long history, dating back to the work of Poincare, Fatou, and Julia [4]. However, the striking beauty and complexity of the resulting shapes in the complex plane was not revealed until the recent investigations of Mandelbrot [7].

The mathematical study of these shapes has led us to consider their counterparts in three- and four-dimensional space. In some cases a 1-parameter family of planar fractals may be better understood by regarding the whole family as a fractal object in 3-space. The major impetus for this work however was the discovery of a class of geometric shapes which result in three and four dimensions from iteration of algebraic operations in the quaternions [9]. Such shapes can properly be visualized only by producing images as close as possible to the 3-D visual experience. With the use of high-resolution computer graphics one simulates the effect of viewing such fractal objects as if they were modeled from clay and illuminated from outside.

This article describes a system developed for generating and displaying such shapes in 3-D. The techniques discussed are generally applicable in the generation and display of connected surfaces which form the boundaries between regions defined by different arithmetic or logical conditions. That is, the surface in

question is required to have an inside and an outside, and it is assumed possible to distinguish in which set a given point lies by performing a series of calculations on the coordinates of the point. The techniques discussed may be used to determine and display any such surface. However, the most interesting applications are on surfaces whose determination requires a great deal of calculation, and the display techniques were particularly designed for very irregular fractal surfaces.

The mathematical principle behind the generation of these fractal shapes involves the iteration of algebraic transformations. Consider a mapping

$$T: \mathbf{R}^n \rightarrow \mathbf{R}^n$$

of n-dimensional Euclidean space into itself. If x is a point in $\mathbf{R}^n$, successive applications of T define a sequence of points in $\mathbf{R}^n$:

$$x_0=x,\ x_1=T(x),\ x_2=T(T(x)),\ x_3=T(T(T(x))),\ \ldots$$

There are several possibilities for such a sequence; for example the points may diverge to infinity; the sequence of points could converge to a finite limit; the sequence could repeat a cyclic series of points; and so on. The result of such an iteration ordinarily depends on the starting point x. When that is the case, points in $\mathbf{R}^n$ may be classified according to the results of such an iteration. For example, $T(x) = x^2$ defines a mapping of the complex plane into itself. If $|x|>1$, successive applications of T result in a sequence tending to infinity. If $|x|<1$ the sequence converges to 0. (Zero and infinity comprise the attractor set of T.) If x is in the circle $|x|=1$, the sequence remains in that circle. In this case the three sets defined by $|x|<1$, $|x|=1$, and $|x|>1$ are each preserved by T. One in fact gains considerable understanding of the transformation T by considering only what it does to the invariant set $|x|=1$; The algebraic symmetries present in the formula for T are exhibited in the geometry of this invariant set. This set can be described as the boundary between the points attracted to zero and those attracted to infinity.

The surfaces illustrated in this article are all defined by analogous phenomena, but where the transformation T may be given on $\mathbf{R}^3$ or $\mathbf{R}^4$ by any series of algebraic operations on the underlying real coordinates of the points. The algebraic symmetries involved in the transformations can be more complex than in the example above and the resulting invariant surfaces usually are fractals. To check whether a given point is inside or outside a given invariant surface, one calculates as many as 1000 iterates in the sequence T(x),T(T(x)),T(T(T(x))),..., testing whether the points satisfy the appropriate conditions.

Fractal dimension and surface modeling

We refer the reader to [6,8] for a thorough discussion of fractals. Fractals are defined (see [6,8]) as mathematical objects whose topological dimension differs from their Hausdorff (fractal) dimension. This dimension is also related to the computer-graphic display of complex objects. Suppose given a complex real-world environment which is to be displayed by computer graphics; e.g., a complex mechanical device or a natural scene. Such an environment must be approximated when it is represented internally in the computer. Typically one can specify a resolution or tolerance and then represent such an environment in terms of primitive objects which are no smaller than the specified tolerance. An estimate of the complexity of such an environment is given by the number of such primitives required at a given resolution. See [11] for a discussion of how various hidden-surface algorithms perform as a function of this type of complexity.

More generally, one may ask how the number of primitive elements varies as the tolerance is decreased. If for example a surface is represented by planar polygons, one would expect the number of such primitives to increase by at least a factor of four when the resolution is doubled. The fractal dimension of the environment can be related to this rate of increase. Suppose the surface of the environment is represented by a set of cubes of a given size (tolerance) which contains the surface. Let N(r) denote the number of such cubes of diameter r which are required. If N(r) grows as a power r^{-D} of the diameter, then D is the fractal dimension of the surface in question. Note that D will generally be between 2 and 3. The fractal dimension is therefore directly related to the number of sample points required to describe the boundary of an object. The boundary representation used here is an approximation of the above cubic description. The rate of growth of the size of the surface models may be used to give a rough approximation of the fractal dimension. Other representations, using for instance planar polygons, can be similarly related to the fractal dimension.

Another measure of the difficulty of graphically representing a complex environment is given by the depth complexity associated with a given viewer position and line of sight. The depth complexity is the number of visible surfaces intersecting the ray from the viewer in the direction of his line of sight. Depth complexity may be considered an indicator of the sorting or priority-checking which is required to determine visibility in a hidden-surface algorithm. Typically a mathematical object in $\mathbf{R}^3$ of dimension D intersects a straight line in a set of dimension D-2. Thus the number of surfaces with which the hidden surface algorithm must deal is of the order of r^{2-D}, depending on the diameter r of primitives.

Surface Determination

The basic technique of calculating these invariant surfaces involves iterating a function repeatedly and keeping track of the points which satisfy certain criteria after many iterations. Generally the result of such a calculation is a determination of whether the starting point is interior or exterior to a specified invariant set. When such a starting point is interior, it is presumed (for purposes of approximation) that the entire cube associated with the point is interior. Nevertheless, these "point determinations" provide only a sampling of the surface and one must maintain a distinction between grid points and the volumes they correspond to. The surfaces could in principle be determined by evaluating every point on a large 3-dimensional grid, then selecting boundary points according to known techniques (see [1,5].) However, the amount of calculation and data involved would prohibit such a determination on a large grid (say 1000x1000x1000). Instead, the surface is followed throughout the grid, without calculating points far from the surface. The interior points which adjoin exterior points are retained (as proved boundary points) and their neighbors become candidates for the boundary, to be tested in the next cycle. The number of function evaluations is reduced considerably by this technique. (Exactly how many function evaluations depends on the fractal dimension of the surface in question.) This technique also provides information about connectivity of the surface: One can separately calculate and examine different connected components of a given surface.

An array processor was employed to do floating-point arithmetic offline. Because the point evaluations can be done by repeatedly performing the same operations on lists of real numbers, the array processor is ideally suited for this type of application. However, these arithmetic calculations still require much more computer time than any other aspect of the surface determination. The array processor was programmed to input a list of up to 2000 points and output a list of as many codes, indicating whether each point is inside or outside the given geometric shape. This process is most efficiently performed when the number of input points approaches its maximum. Therefore, in following the surface, it was important to deal efficiently with large stacks, rather than keeping the stacks short. What is presented here is a systematic way of effectively keeping track of the results of calculations, and ensuring that most of the computer time involved is spent on function evaluation, not on manipulation of points. These methods were applied on an IBM 3033 with a Floating Point Systems AP190L array processor. Ordinarily the function evaluation on the array processor used more than 90% of the total CPU time.

The basic requirements of the surface determination are as follows:

1. Repetition of point evaluations should be avoided, as this is the most time-consuming operation.
2. The algorithm should make no assumptions about the extent of the surface; it may be extremely convoluted (even approaching space filling) or it may be very smooth.
3. No sorting or comparison of stacks should be done; all stack manipulations should be linear in complexity, so that there be no penalty for using large stacks.

The final output of this process is a list of points P (boundary points) on the grid, satisfying the following:

1. P is inside the shape in question
2. At least one of the neighbors of P is outside the shape in question.
3. P is connected to one of a set of starting points via a path which follows the grid and consists only of boundary points.

Note that the above 3 conditions provide a recursive definition of the desired set of boundary points, depending only on the mathematical shape in question, the grid used, and the set of starting points. To determine all boundary points, maintain a stack of newly-discovered boundary points, and perform the following until this stack is exhausted:

1. List all neighbors of newly-discovered boundary points.
2. Determine which of these neighbors are interior. Those which are interior are candidates for the boundary (since they satisfy conditions 1 and 3 above).
3. Check the neighbors of the candidates. If a candidate has a neighbor which is exterior, the candidate is a boundary point, and is listed with the newly-discovered boundary points.

In order to be precise, one must define the notion of adjacency which is to be used in determining the "neighbors" found in steps 1 and 3. It was in fact useful to use different notions of adjacency, depending on the fractal characteristics of the surface being investigated. The notion of adjacency used in step 1 determines the extent of the surface; therefore surfaces which contain long, thin strands are more thoroughly delineated by using a liberal notion of adjacency, for instance considering cubes which touch only at a corner to be adjacent. The notion of adjacency used in step 3 determines how thoroughly the surface is to be covered. In deciding whether a given interior cube was on the boundary, it was usually required to share a face with an exterior cube.

In order to begin the surface determination, one must specify a starting list of boundary points. (The choice of starting points will establish which components of the surface are to be determined.) The starting

points can be easily specified by choosing a line segment which joins the interior and exterior of the given shape. At least one point on such a segment will be a boundary point, and can serve as a "seed" for the surface determination.

It should be observed that the names "interior" and "exterior" are chosen only to distinguish two alternative properties, not necessarily describing bounded and unbounded sets. In fact, the algorithm allows one to follow either the interior or exterior of the given mathematical boundary. There are often interesting qualitative differences between these two sets, as shown in the illustrations.

Display Methods

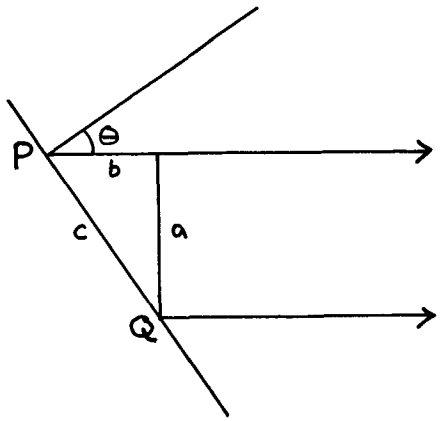
The output of the preceding calculation is a sorted file of grid vertices; typically such a file will contain over one million points. This list of grid points provides at best inner and outer limits to the extent of the surface; at worst a biased sampling of points near the surface. To each point on the list corresponds a cube which could be regarded as the primitive for the display process. However, one only can conclude that the cube in question intersects the given volume, not that the cube's surface coincides with part of the fractal surface being displayed.

Because the surfaces to be displayed are fractal, the display techniques were chosen so as to avoid the customary preference for smoothness. The details one sees at a given resolution should suggest higher resolution will reveal more detail. It was therefore considered inappropriate to model the surfaces with planar or smooth primitives; the points themselves become the primitives of the display process. Conceptually the presence of a point on the list represents only the fact that a certain portion of 3-space is occupied by some opaque material; this should not imply the presence of a smooth surface bounding that opaque material.

The display process is in two stages. In the first stage, illumination intensities are assigned to each vertex, depending on the relationship of the vertices to an imagined light source. During the second stage, an image is produced, depending on the direction to the viewer. The first stage is more time-consuming, requiring two passes through the data points. The second, image-producing stage, involves only one sequential pass through the data. The second stage may be performed repeatedly, to obtain a series of views of the object from different angles. The algorithms for both processes are based on z-buffers; that is, the model is projected into a two-dimensional buffer in memory, the projection being performed in such an order so as to handle shading and hidden surface elimination. Because of the simplicity of the primitive display elements, no sorting is required for either illumination and display projections. If the data initially is sorted on increasing model coordinates, then the z-buffer projection can for most directions be achieved by a forward or reverse pass through the data. Projection in the remaining directions can be achieved by a simple restructuring of the initial sort, again done in linear time. Alternatively, the visible or illuminated surfaces can be found by determining the distance of each point from the viewer or light-source at the time of the projection, retaining closest points. This also is done in linear time, but may require additional space to retain visible z-coordinates.

The use of z-buffers makes this an image-space rather than object-space algorithm; this is not a serious drawback, because the surfaces are only calculated to a prespecified accuracy. The primary difficulty encountered with z-buffer techniques was in determining the illumination of surfaces which are nearly parallel to the light direction. This is a situation where some of the information ordinarily lost in a z-buffer projection must be recovered. Our solution to this problem was to regard the surface cubes as somewhat translucent; cubes immediately behind illuminated cubes receive a portion of the light of the illuminated ones. This technique is particularly useful for eliminating staircasing when the light direction is along one of the grid axes.

Fractal surfaces are not differentiable. This implies that the surface normal vector at a given point is generally not defined. In order to simulate light reflectance one can however determine a "typical" normal direction at a given point by comparing its coordinates with those of nearby points. To obtain an approximate normal vector at a given illuminated point (which has projected into the z-buffer), the z-coordinates of its neighbors in the z-buffer are compared. The difference in z-coordinate between two points divided by their separation in the z-buffer is used to estimate the tangent of the angle between the light source and the surface normal. The average of several such surface-normal approximations can be used to define the reflectivity at a point. The figure below shows how such a normal vector is calculated:



Points labeled P and Q are projected to a z-buffer in the direction indicated by parallel arrows. The line perpendicular to PQ forms an angle theta with the illumination direction. Cos(θ) may be calculated as the ratio a/c, where a is proportional to the distance between the projections of P and Q in the z-buffer, b is the difference in z-coordinates of the two points, and c is the corresponding hypotenuse. Two of these calculations at right angles determine a normal vector.

The final part of the picture display consists of projecting the points in the direction of an imagined viewer, using the intensities determined above to decide how bright a given point will appear. this process is a straightforward z-buffer algorithm and is performed with an orthogonal projection. In order to ensure that the visible surfaces completely conceal the hidden surfaces, individual points in the model may be projected to several nearby positions in the image z-buffer. The result is a half-tone picture in which each visible point in the model produces one or more dots on the picture.

Illustrations

The illustrations have been chosen as examples of the diversity of form and texture which result from these techniques. The first two illustrations show 1-parameter families of fractals in the complex plane. Each horizontal slice is the shape in the complex plane corresponding to choosing one value of the parameter, and the whole figure shows the effect of changing the parameter. Shapes in the complex plane upon which these figures were based were first displayed graphically by B. Mandelbrot. See plate 187 in [8] for another illustration of this type. The surfaces in figures 1 and 2 are defined as follows: For each complex number λ of absolute value 1, iterate the transformation $T(z) = \lambda z(1-z)$. The set of complex numbers z for which the successive iterates do not tend to infinity defines a subset of the complex plane. The illustrated 3-d shapes show how this planar shape varies for λ on the circle $|\lambda| = 1$. By a change of variables, the shape in figure 1 is "unwound", producing figure 2. The two objects were followed on grids of size 600^3. Each is represented by approximately two million points. They are displayed at resolution 600x800 using two light sources.

Figures 3 through 6 illustrate some invariant shapes determined by the iteration of quadratic polynomials in the four-dimensional quaternions [3]. (See [9] for a discussion of the mathematics involved in iteration of rational functions in the quaternions). For our present purposes, it suffices to observe that quaternion multiplication and addition provide a 4-dimensional generalization of complex arithmetic, based on vector algebra. Models in $\mathbf{R}^3$ are produced by starting the iteration in a three dimensional subspace of the quaternions spanned by 1, **i**, and **j**. The resulting shapes were followed on a grid of size 1200^3. They were displayed at resolution 1024x1280 using two light sources.

Figure 3 is defined by iteration of a quadratic polynomial in the quaternions, of the form $\lambda z(1-z)$, with a complex number λ. The parameter λ was chosen so as to obtain an attractive cycle of length 7, using techniques developed in [7]. The shape shown is a connected component of the set of points attracted to that cycle. The shape is invariant under a 7-fold iteration of the defining transformation, and is one of an infinite number of components defined by that invariance. This component is represented by about 5.6 million points.

Figure 4 is defined by the iteration of another polynomial of the form $\lambda z(1-z)$. In this case, the shapes illustrated are different components of the set attracted to a cycle of length four.

Figure 5 is defined by the iteration of a polynomial of the form $z^2+\mu$, with μ a complex number chosen so as to produce an attractive cycle of length 4. The illustrated shape is the boundary of the set attracted to that cycle. Four colors are used to distinguish invariant sets defined by 4-fold iteration of the transformation.

Figure 6 is defined by the polynomial $\mathbf{i}(z^2+1)$. The illustrated shape is one of two components which together comprise the set of points attracted to a cycle of length two.

Acknowledgments

I am indebted to Benoit Mandelbrot for advice and encouragement, without which this work could not have been done. I also benefited greatly from conversations with Douglas McKenna and Richard Voss.

References

1. Artzy, E., Friedan, G., and Herman, G., The Theory, Design, Implementation and Evaluation of a Three-dimensional Surface-Detection Algorithm, **Comp. Graph. & Im. Proc.**, 15, 1981, pp. 1-24.

2. Carpenter, L.C., Fournier, A., and Fussel, D., Display of fractal curves and surfaces, to appear, Comm. ACM.

3. Hamilton, Sir W. R., **Elements of Quaternions**, Vols. I and II, Reprinted by Chelsea Publ. Co., New York, 1969.

4. Julia, G., Mémoire sur l'iteration des fonctions rationnelles, J. Math. Pure Appl. **4**:47-245, 1918. Reprinted in **Oeuvres de Gaston Julia**, Vol. I, Gautier-Villars, Paris, 1968.

5. Lin, H. K., Two and Three Dimensional Boundary Detection, **Comp. Graph. and Im. Proc.**, 6, 1977, pp. 123-134.

6. Mandelbrot, B. B., **Fractals: Form, Chance, and Dimension**, Freeman, San Francisco, 1977.

7. Mandelbrot, B. B., Fractal Aspects of the Iteration of $z \rightarrow \lambda z(1-z)$ for complex λ and z, **Ann. N. Y. Acad. Sci.** 357, 1980, pp. 249-259.

8. Mandelbrot, B. B., **The Fractal Geometry of Nature**, Freeman, San Francisco, 1982.

9. Mandelbrot, B. B., and Norton, A., Fractal Surfaces Defined by Iteration of Rational Functions in the Quaternions, to appear.

10. Newman, W. M., and Sproull, R. F., **Principles of Interactive Computer Graphics**, McGraw-Hill, New York, 1973.

11. Sutherland, I., Sproull, R., and Schumacker, R., A Characterization of Ten Hidden-Surface Algorithms, **Computing Surveys**, Vol. 6, No. 1, 1974.

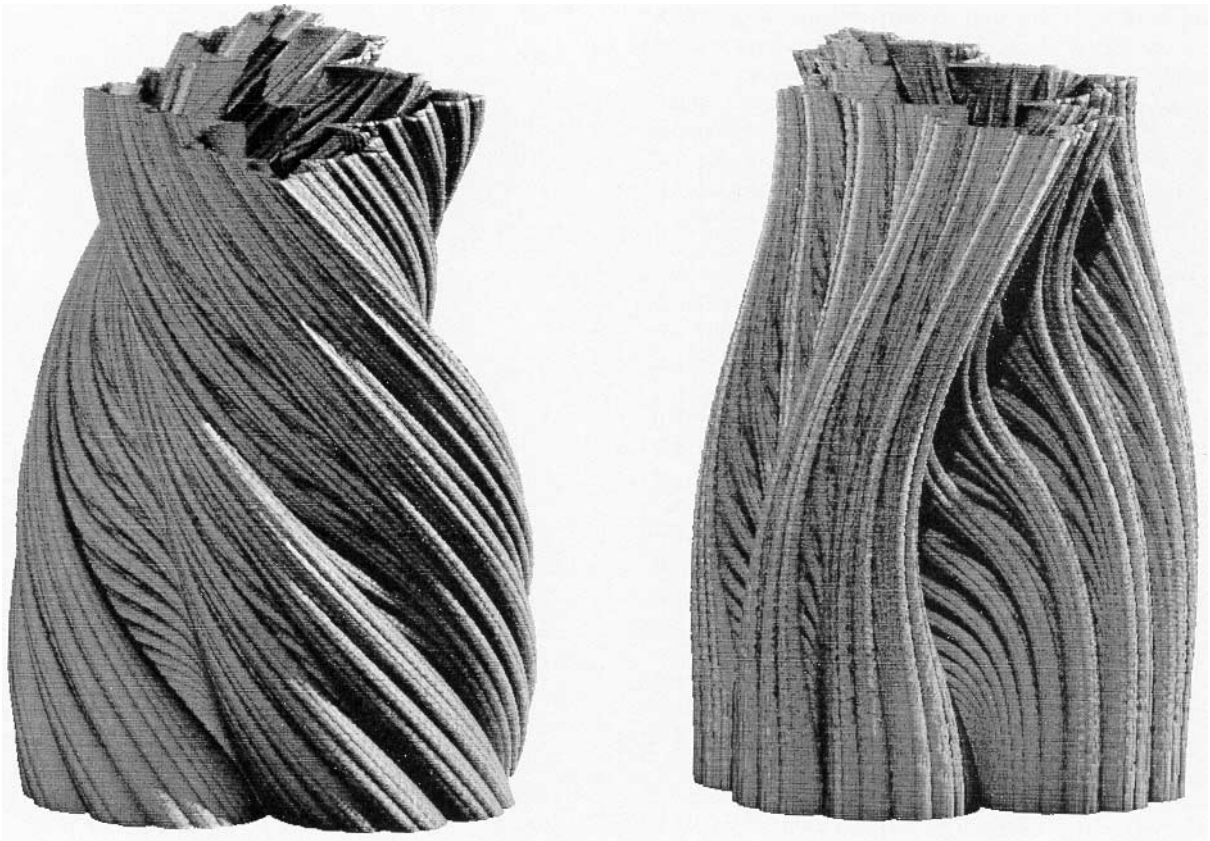


Figure 1 Figure 2

Two 3-d objects defined as 1-parameter families of fractal curves.
Each horizontal slice results from iterating a quadratic polynomial in the complex plane.

Figure 3

A fractal surface invariant under 7-fold iteration of a polynomial in the quaternions.

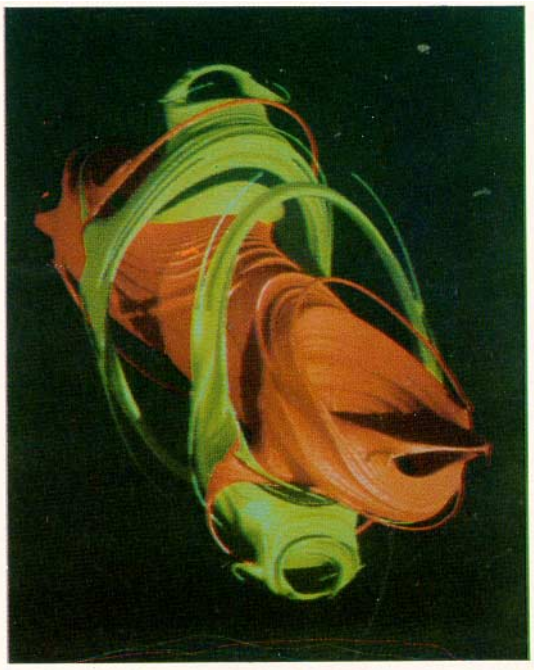


Figure 4

Several components of the domain attracted to a cycle of length 4.

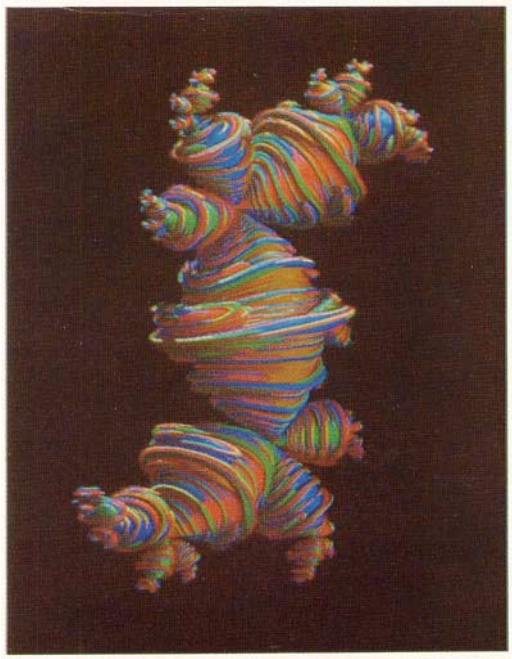


Figure 5

Many components of the domain attracted to a cycle of length 4.

Figure 6

One of 2 components of the shape defined by $i(x^2+1)$.