FRACTAL LANDSCAPE SYNTHESIS IN COMPUTER GRAPHICS

by

Jay S. Gondek

A THESIS

Presented to the Department of Computer
and Information Science
of the University of Oregon
in partial fulfillment of the requirements
for the Departmental Honors Program

April 1992

Abstract

Traditional computer graphics modelling techniques have failed to capture the complexity of natural topography. Solutions to this problem have been found by using fractal geometry. This paper provides an introduction to fractal geometry, including fractal algorithms, dimension, and self-similarity. Two distinct categories of fractals, deterministic and random, are compared. The random fractal, and in particular, the 1/f-noise model of fractional Brownian motion (fBm), simulates natural terrain. Three different methods of synthesizing fBm are investigated: midpoint displacement, random faults, and Fourier analysis. Images of synthesized topography are presented to aid in visualizing the algorithmic steps and final results of each procedure. The attributes of each method are explained, and suggestions for applying the techniques to particular rendering problems are given.

Contents

List of Color Plates

List of Figures

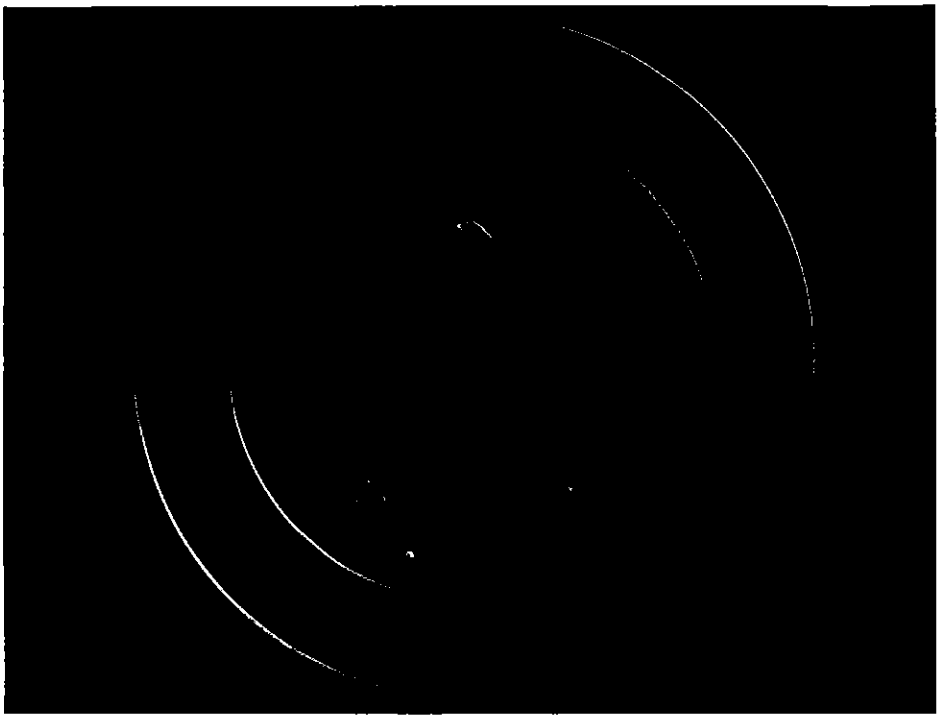


Plate 1: Mt. Hood in Depiction of U of O Seal

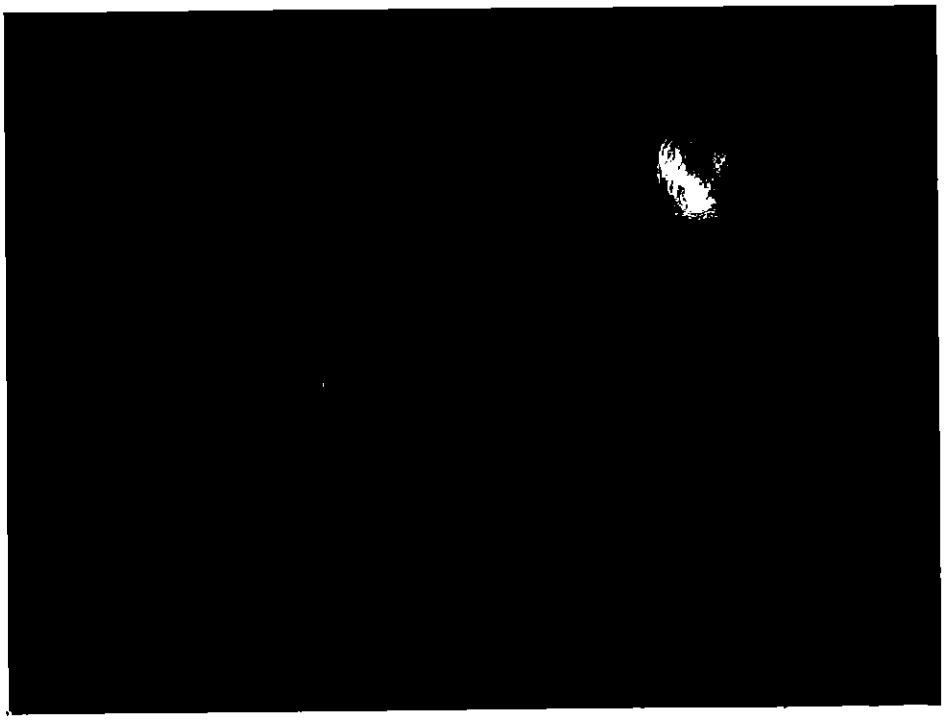


Plate 2: Simulation of an Earth-like Planet

Introduction

Geometric modeling is an important area of computer graphics that greatly aids industrial design and visualization. Creating a computer-based model requires specifying Euclidean shapes and smooth surfaces, and carefully piecing those shapes together. Traditional design techniques have been instrumental in the creation of complex manufactured items, ranging from buildings to combustion engines. Although Euclidean shapes can be used to describe complex industrial items, they fail to accurately describe natural phenomena such as clouds, mountains, coastlines, and oceans. However, the unlimited detail of nature can be simulated using fractal geometry.

Fractal geometry, a relatively new branch of mathematics, pertains to the study of infinitely complex numerical sets. Despite their intricacy, fractals can be fully described with concise sequences of iterated steps. Because fractal descriptions are algorithmic, the advent of the computer greatly extended our ability to calculate and visualize fractal constructs. Researchers in the field of computer graphics began to see the potential of applying fractal algorithms to inherently difficult modelling problems. One formidable problem is the synthesis of natural phenomena. To numerically describe a natural scene, the overwhelming amount of detail required rules out the application of standard geometry. Fractal algorithms can generate this high level of detail, and in particular, a fractal model was developed that elegantly solves the problem of synthesizing naturally occurring rough edges and surfaces. The focus of this thesis is the fractal surface model and its implementation.

This thesis begins with a general section about fractal geometry. In this section, the term "fractal" is defined, and fractal properties are discussed. The model that produces rough surfaces, termed fractional Brownian motion, is quantified. The next section is devoted to examining implementations of fractional Brownian motion. This part begins with a practical discussion of landscape rendering. The following subsection introduces the "midpoint displacement" technique for modelling fractal terrain. Several variations of this method are discussed, and examples are illustrated with computer generated images. The next subsection examines the "random faults" method, a procedure that can be used to simulate planetary relief. Finally, a terrain synthesis scheme is described that uses the Fourier transform to strictly adhere to the fractional Brownian motion paradigm. The conclusion compares the three implementations of fractional Brownian motion, and this thesis closes with suggestions about future applications of random fractal surfaces.

Fractal Geometry

An informal definition of a fractal is "a geometric construction that exhibits detail at all scales." Thus, viewing a fractal under increasing powers of magnification will not reduce the visible detail, as in Euclidean geometry, but will display an equal amount of complexity at each level [10]. Fractals are computed with a recursive or iterative procedure. The limit of the iteration is the fractal itself. For example, Sierpinski's Triangle is produced by starting with a set A containing a single closed-

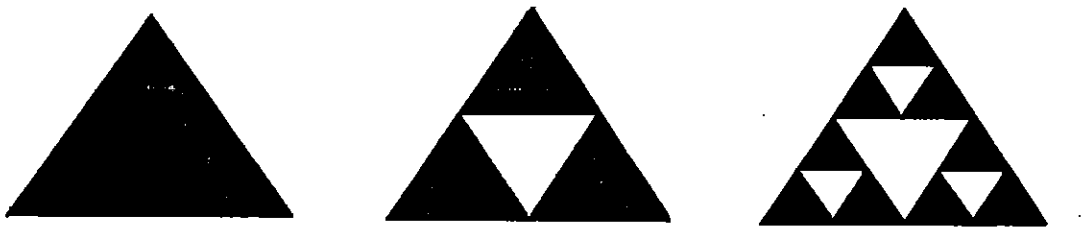


Figure 1: Construction of Sierpinski's Triangle

interval two-dimensional equilateral triangle. The iterative step, shown in Figure 1, is the following:

> For each (two-dimensional) triangle in set A, remove a section that is a 1/4 area inverted open-interval equilateral triangle.

At the limit of this iteration, all area of the triangle has been removed, but because only open-intervals were removed from the original closed-interval triangle, the set will be a fully connected infinitely thin and long web known as Sierpinski's triangle (Figure 2). This example gives a casual look at fractal geometry. A more formal approach involves analyzing the dimension of fractal sets.

Fractal Dimension

Benoit Mandelbrot, a pioneer in the field of fractal geometry, coined the word "fractal" and defined it to mean "a set for which the Hausdorff-Besicovitch [fractional]

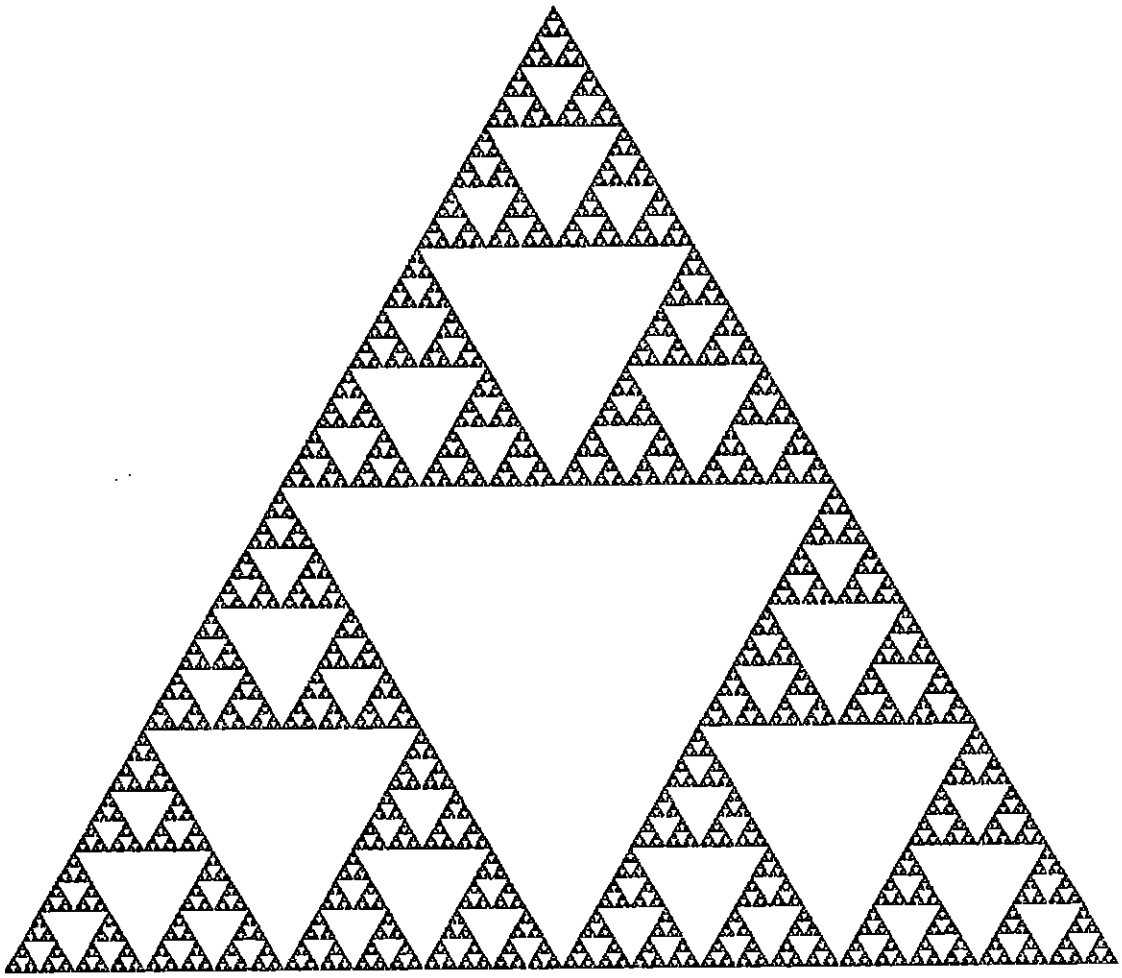


Figure 2: Sierpinski's Triangle

dimension strictly exceeds the topological dimension" [4]. Most fractals are infinite in size in their topological dimension, and zero in size in the next higher topological dimension. For example, Sierpinski's triangle has a topological dimension of one. However, measuring the length of Sierpinski's triangle (hereafter referred to as S) produces an infinite result. If S is measured in the second dimension, the result is zero, because it does not fill any area in the two-dimensional plane. It seems that the fractal dimension of S lies between one and two.

Intuitively, the fractal dimension of an arbitrary fractal A is the dimension for which A exhibits a finite non-zero measure. The Hausdorff-Besicovitch procedure for determining fractal dimension measures A within an arbitrary dimension p, producing the Hausdorff p-dimensional measure of A. One can then determine the value of p that allows the Hausdorff p-dimensional measure of A to take on a finite non-zero value, producing the Hausdorff-Besicovitch dimension of A [1]. For example, S has a fractal dimension of $\ln(3)/\ln(2)$, which is between one and two, as predicted. The Hausdorff p-dimensional measure of S can be determined by calculating the number of p-dimensional spheres that provide a minimum cover over S. Sieradski [12] explained an easy way of inductively determining the Hausdorff p-measure, which involves examining the constructive steps used to produce a fractal. Figure 1 displays the first three constructive steps used to produce S. Assuming the diameter of the bounding sphere of the triangle in Figure 1a is δ, Table 1 shows a method of producing the general relationship between the iterative step, and the number and diameter of minimal-cover bounding spheres for S. At any iteration level, the set S is contained

level	*diameter*	*spheres*
S_0 (Figure 1a)	$\delta(1/2)^0$	1
S_1 (Figure 1b)	$\delta(1/2)^1$	3
S_2 (Figure 1c)	$\delta(1/2)^2$	9
$\vdots$	$\vdots$	$\vdots$
S_k (Figure 2)	$\delta(1/2)^k$	3^k

Table 1: Bounding Limits for Sierpinski's Triangle

within a plane, and thus a circle can be used as the bounding sphere. If one circle of diameter δ provides a minimum cover over Figure 1a, then three circles of diameter $\delta/2$ provide a minimum cover over Figure 1b. In general, it can be reasoned that 3^k circles of diameter $\delta(1/2)^k$ give a minimum cover for iteration level k. These values can be used to compute the fractal dimension and size of S:

$$p\text{-measure} = \lim_{k\to\infty} 3^k \left[\delta\left(\frac{1}{2}\right)^k\right]^p$$

$$= \delta^p \lim_{k\to\infty} \left(\frac{3^k}{2^{kp}}\right)$$

$$= \delta^p \lim_{k\to\infty} \left(\frac{3}{2^p}\right)^k$$

To produce a non-zero finite result:

$$\left(\frac{3}{2^p}\right) = 1$$

Thus p, the Hausdorff-Besicovitch dimension of S, is:

$$p = \frac{\ln 3}{\ln 2}$$

and the "size" of S is:

$$\text{Hausdorff } p\text{-measure} = \begin{cases} \infty & \text{if } p < \frac{\ln 3}{\ln 2} \\ \delta^{\ln 3/\ln 2} & \text{if } p = \frac{\ln 3}{\ln 2} \\ 0 & \text{if } p > \frac{\ln 3}{\ln 2} \end{cases}$$

Fractal Classes

Sierpinski's triangle is a good introduction to fractals, but the fractals that simulate nature belong to an entirely different category. There are two major classes of fractals: deterministic and nondeterministic. In the case of a deterministic fractal, the iterative function producing the fractal will always give the same result. For instance, Sierpinski's triangle is deterministic; its shape is invariant. Nondeterministic fractal algorithms rely on input from a random number function to produce a random fractal set [8].

Ideally, a nondeterministic, or random, fractal algorithm will always produce different results. In practice, random numbers are generated by pseudo-random number functions. If two identical instances of a particular pseudo-random number generator are initialized with the same parameters, they will produce identical output. Varying the parameters of the implemented pseudo- random number function allows a single algorithm to output many different results. Another difference between random and deterministic fractals is how the fractal property of self-similarity manifests.

Random fractals are not self-similar in the way that many deterministic fractals are. Subsections of a random fractal are not identical to the whole; they cannot

be scaled and overlapped onto the whole as was the case with Sierpinski's triangle. Because random fractals are the result of a stochastic process, their properties can be best quantified using statistics. Although random fractals are not self-similar, sections of a random fractal have the same statistical properties of the whole, "modulo a vertical scaling factor" [8]. This property is known as "statistical self-affinity".

Fractional Brownian Motion

Many random fractal constructs that closely resemble the rough fractal surfaces seen in nature are based on fractional Brownian motion (fBm). FBm, if viewed as a waveform, has the property that the expected value of the mean-square fluctuation in amplitude at frequency f is proportional to $1/f^{\beta}$, where β is a constant that describes the roughness of the Brownian curve [9]. This inverse power law simply means that points that are close in space (sampled at a high frequency) are going to have relatively similar amplitudes. Points that are sampled at a low frequency may be far apart in amplitude. This model is evident in many surfaces formed in nature. For example, if elevation is substituted for amplitude, this model approximates change in elevation over distance in a mountain range. Why the simple $1/f$- noise paradigm describes many natural systems is a mystery. Richard Voss has done a considerable amount of work with the $1/f$ phenomena, and he explains:

> Little is known about the physical origins of $1/f$, but it is found in many physical systems: in almost all electronic components from simple carbon resistors to vacuum tubes and all semiconducting devices; in all time standards from the most accurate atomic clocks and quartz oscillators to the ancient hourglass; in ocean flows and the changes in yearly flood levels of the river Nile as recorded by the ancient Egyptians; in small voltages

> measurable across nerve membranes due to sodium and potassium flow; and even in the flow of automobiles on an expressway. $1/f$-noise is also found in music. [9]

Clearly there are many interesting modeling applications that could exploit $1/f$-noise. The focus of this thesis is the exploration of $1/f^{\beta}$-noise random fractal models that simulate natural topography.

To approximate natural fractal surfaces such as topography or clouds, the values assigned to β must be limited. Assigning low values to β will produce results that are too rough. As β approaches zero, the resulting surface approaches completely uncorrelated white noise. Conversely, high values of β produce surfaces that are smooth and highly correlated. Voss claims that choosing β from the interval $[1,3]$ works well for simulations of nature. With this restricted range of β, the fractal dimension D can be estimated as $D = E + (3 - \beta)/2$, where E is the topological dimension [9]. For example, as the β associated with a topographic relief varies from one to three, the estimated fractal dimension of the relief will vary from two to three. Thus, random fractal algorithms use β as a control parameter to vary the correlation of elevation points.

Random Fractal Models

Attempts to simulate natural topography have led to several procedures that model fractional Brownian motion. Depending on the specific application, there are a number of features that a fractal procedure must have. One of the most important aspects

of any fBm method is its mathematical correctness. If the terrain generating procedure does not accurately implement the $1/f$-noise model, the results will be flawed. For many applications, the ability to add detail to an existing topography database is essential. If a computer animation is to use fractal terrain, then the dynamics of the viewpoint may require changes in the resolution of the terrain and extensions to the terrain in any direction. The underlying geometry of the fractal surface must also be considered; not all implementations of the fBm paradigm generalize to any surface shape. At this time, there is no particular fBm algorithm that encompasses all of these features. Three random fractal procedures are presented, along with descriptions of their abilities and shortcomings.

Midpoint Displacement

One of the most widely used techniques for generating random fractal surfaces is midpoint displacement, often referred to as recursive subdivision. Many published fractal mountains are the result of a midpoint displacement scheme. This model received notoriety after it was used to produce a planet surface for the Genesis scene of *Star Trek II: The Wrath of Khan* [9]. Loren Carpenter, who was responsible for the Star Trek scene, combined research efforts with Alain Fournier and Don Fussel to publish some of the initial midpoint displacement methods [3].

The Fournier, Fussel, and Carpenter algorithms are based on the recursive subdivision of a surface. The surface to be subdivided is usually represented as a two-dimensional array of height values, referred to as a height field. During the

recursive subdivision, height values are assigned to each point of the height field. The position of any height value in the array can be interpreted as two- dimensional coordinates. The height value provides the third coordinate, and with this data, the surface can be tessellated and displayed in three dimensions.

At the start of a midpoint algorithm, the height values of the four corners of the array are initialized with Gaussian random numbers, and scaled according to a user-defined standard deviation s. New points are approximated by averaging the height of neighbors. Then, the new points are offset by a Gaussian random number scaled according to $s' = s(1/f^{\beta/2})$. Scaling by $1/f^{\beta/2}$ is necessary to provide an expected mean square fluctuation of $1/f^{\beta}$ at frequency f. This process iterates until f exceeds the resolution of the array. With each full iteration of the algorithm, the resolution of the defined fractal doubles. In theory, the limit of iterations will produce an infinitely complex and rough surface. Because the computer is bounded by memory limitations, the size of the array will determine the limit of iteration, and the data generated will represent only an approximation of the fractal.

A example of a recursive subdivision scheme is shown in Figure 3 [3]. After the first full iteration, the height field will have nine defined height values, as represented by black dots in Figure 3a. Figure 3b shows that new data points are interpolated by averaging neighboring points, resulting in the grey dots. The new points are then offset by a Gaussian random number, with the expected value of the offset being proportional to $1/f^{\beta/2}$. As Figure 3c shows, one more pass completely doubles the resolution. Figure 4 illustrates this process with the height values plotted as elevation

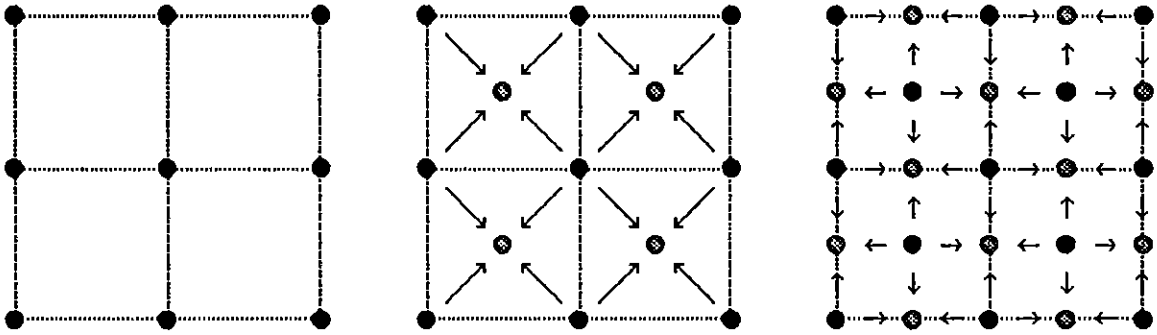


Figure 3: Diamond-Square Subdivision

points of a fractal landscape. Figures 4a, b, c, and d are the result of one, three, five, and seven iterations, respectively. Miller [7] refers to this particular method as "Diamond-Square Subdivision".

Diamond-Square subdivision is a context-dependent process because the weights of neighboring points from all sides are used to estimate new data points. Many popular methods of recursive subdivision are context independent; they do not use all neighboring points to aid in approximating new points during the recursive process [7]. For example, Pokorny and Gerald [11] presented the method of context independent subdivision that is shown in Figure 5. This subdivision technique is more efficient than the Diamond- Square method; it doubles the resolution of the fractal surface with one pass, interpolating the white and grey points from the neighboring black points. However, this technique contains several problems with mathematical

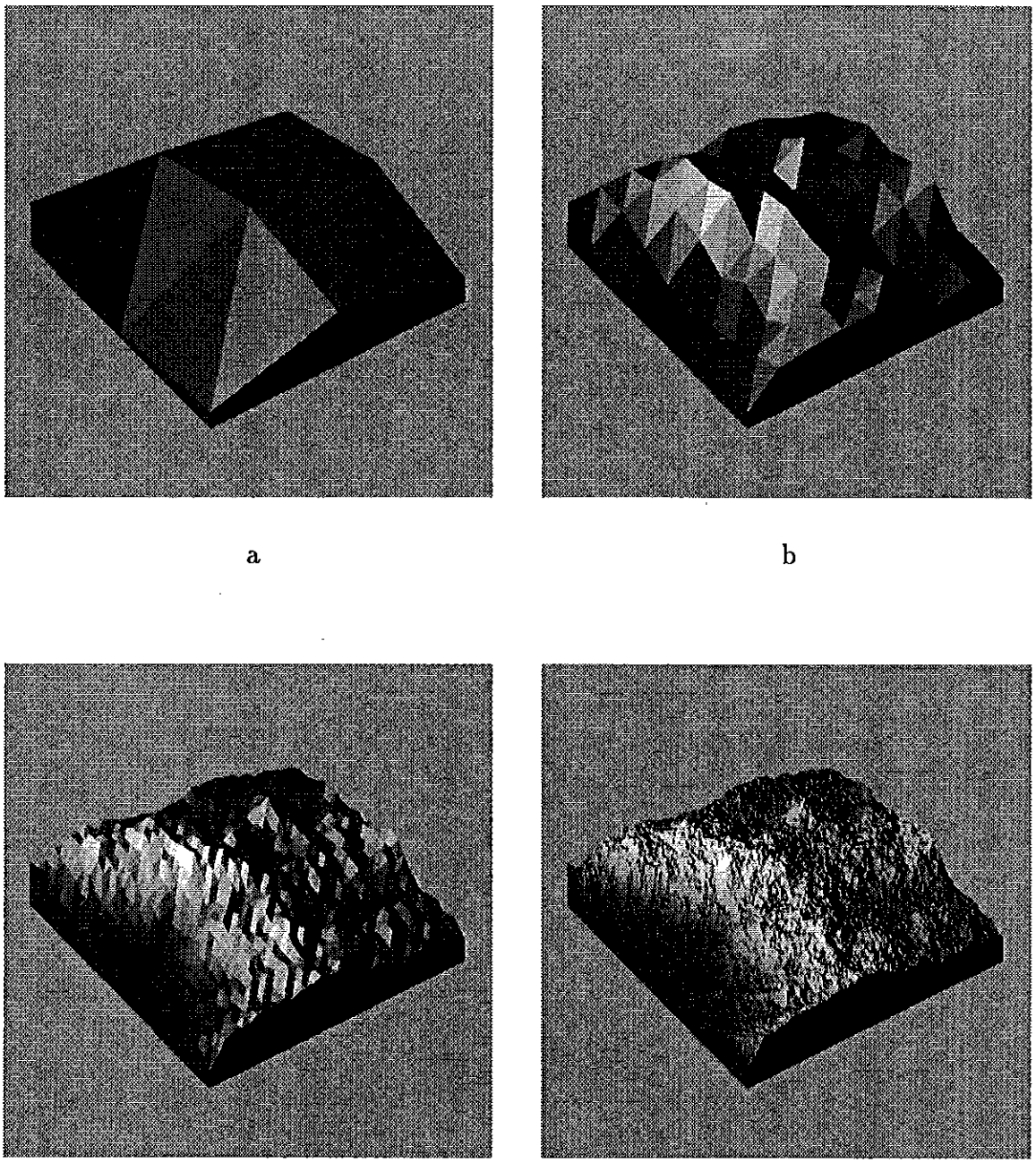
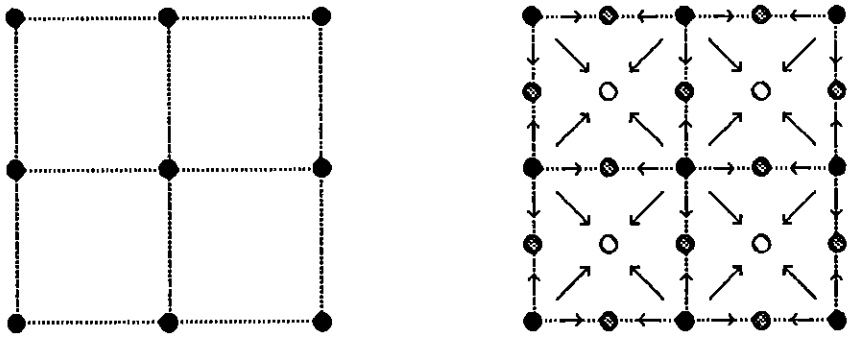


Figure 4: Midpoint Displacement Example
One, three, five, and seven iterations of the diamond-square algorithm are shown.



a b

Figure 5: Context Independent Subdivision

and visual correctness. As shown in Figure 5, the white midpoints are calculated by averaging the four neighboring black points. The grey points result from averaging only two neighbors. Clearly this inconsistency will result in discontinuities because of the lack of correlation between the grey points and half of their neighbors. These regions of discontinuity bisect the elevation data at all levels of recursion and form unsightly creases in three-dimensional rendered height fields [7].

Although the Diamond-Square subdivision scheme has fewer shortcomings than context independent methods, it does not eliminate all artifacts. Using four neighboring points instead of two to estimate new values increases the accuracy of the model, but does not fully correlate all points. Because of this lack of correlation, "creasing" is still apparent, although not as obvious, in the the output. This procedure also exhibits another artifact known as "tenting". Tenting produces pointed

peaks at control points and other other points of relatively high elevation [7]. This occurs when a particular elevation value is much higher than surrounding values, resulting in newly interpolated points that are heavily weighted by the many low points. Relatively low elevation points similarly produce an inverted tenting artifact. Musgrave has shown that midpoint displacement artifacts may sometimes be used to add aesthetic appeal to images. He has used the tenting artifact to give a "storybook castle" look to some of his landscape renderings [8].

Much attention has been given to the visual artifact problems associated with the midpoint displacement techniques. Shortly after the publication of "Computer Rendering of Stochastic Models," where Fournier et al. present several recursive subdivision techniques, Mandelbrot replied to Communications of the ACM denouncing this method [5]. The crux of Mandelbrot's argument was that the mathematical impurity of the recursive subdivision method led to artifacts described as "peculiar 'crumpled paper' texture and a very conspicuous grid of parallel lines along relief breaks". Although recursive subdivision generates conspicuous artifacts, it is one of the fastest random fractal techniques available, and thus cannot be ignored.

Several attempts have been made to ameliorate the artifact problem. Miller [7] proposed a new method of subdivision, which he calls "Square-Square Subdivision". Miller's square-square procedure was adapted from computer aided design methods and generates a fractal surface that is biquadratic (continuous in the first derivitive, and thus insuring a continuous slope). Although this scheme eliminates artifacts that are inherent in other methods, it has two major drawbacks: the generated surface does

not pass through user defined control points, and because the surface is biquadratic, it lacks the fine-scale rough appearance that is visible in other models. Another approach, suggested by Mandelbrot, uses hexagons instead of squares as the geometric shape of the "tiles" used in the recursive subdivision [9]. According to Mandelbrot, because hexagons cannot be exactly recursively subdivided (i.e., precisely filled with smaller hexagons), the boundaries between tiles of subdivision will overlap, transforming the straight-line creasing evident in other models to a more natural looking "crumpled" curve. Voss proposed a method of reducing artifacts, called "successive random additions," that adds $1/f$-scaled random displacement to the entire height field at each level of recursion [9]. This filter can be applied to any midpoint displacement scheme, and has the effect of disrupting the creasing patterns that are formed during subdivision. With several methods available to eliminate artifacts, the midpoint displacement methods are viable options for the production of random fractal surfaces.

One strong advantage of the midpoint displacement technique is that the height field can be initially seeded with user-defined values. As the iterations of the algorithm increase the data resolution, the general form defined by the initial data is maintained. For example, if one point in the center of the height field is initially seeded with a high elevation value, as the algorithm runs, this seed will have the effect of increasing the elevation of points generated around it. These additional points, in turn, influence the elevation of neighboring points. At the end of the iterations, the one user defined point will be a mountain peak, having influenced all

surrounding points to, in general, form a slope to the peak.

The method of seeding the height field and then applying a midpoint displacement algorithm was used to generate a photo- realistic image of Mt. Hood, as seen in Plate 1 and Figure 6. The initial height field was seeded with 1000 elevation points, taken from a contour map. Clearly, if only these initial 1000 elevation points were tessellated and rendered, the size of the polygons defining the surface would be relatively large, giving the mountain an artificial geometric appearance. The 1000 elevation points were uniformly distributed over a height field of much higher resolution. Applying the diamond-square midpoint algorithm to this seeded height field produced over one million data points, adding random, rough detail to the image, but maintaining the shape defined by the points from the contour map. The fractal-generated height field was then overlaid onto a larger height field, and low-land terrain surrounding the mountain was randomly generated and smoothly fused with the mountain data. The "creasing" artifact was almost completely eliminated by using Voss's successive random additions method. The "tenting" artifact was controlled by passing the fractal data through a filter that adjusted any data points that deviated from their neighbors by a significant amount. Part of the surrounding terrain was seeded to produce a low elevation area to be "filled" with water, as seen in Plate 1. The water surface was created by using a relatively smooth random fractal height field as control data for a bicubic spline surface.

The ability to set control points that influence the outcome of the midpoint algorithm can be one of the most valuable features of this technique. This model also

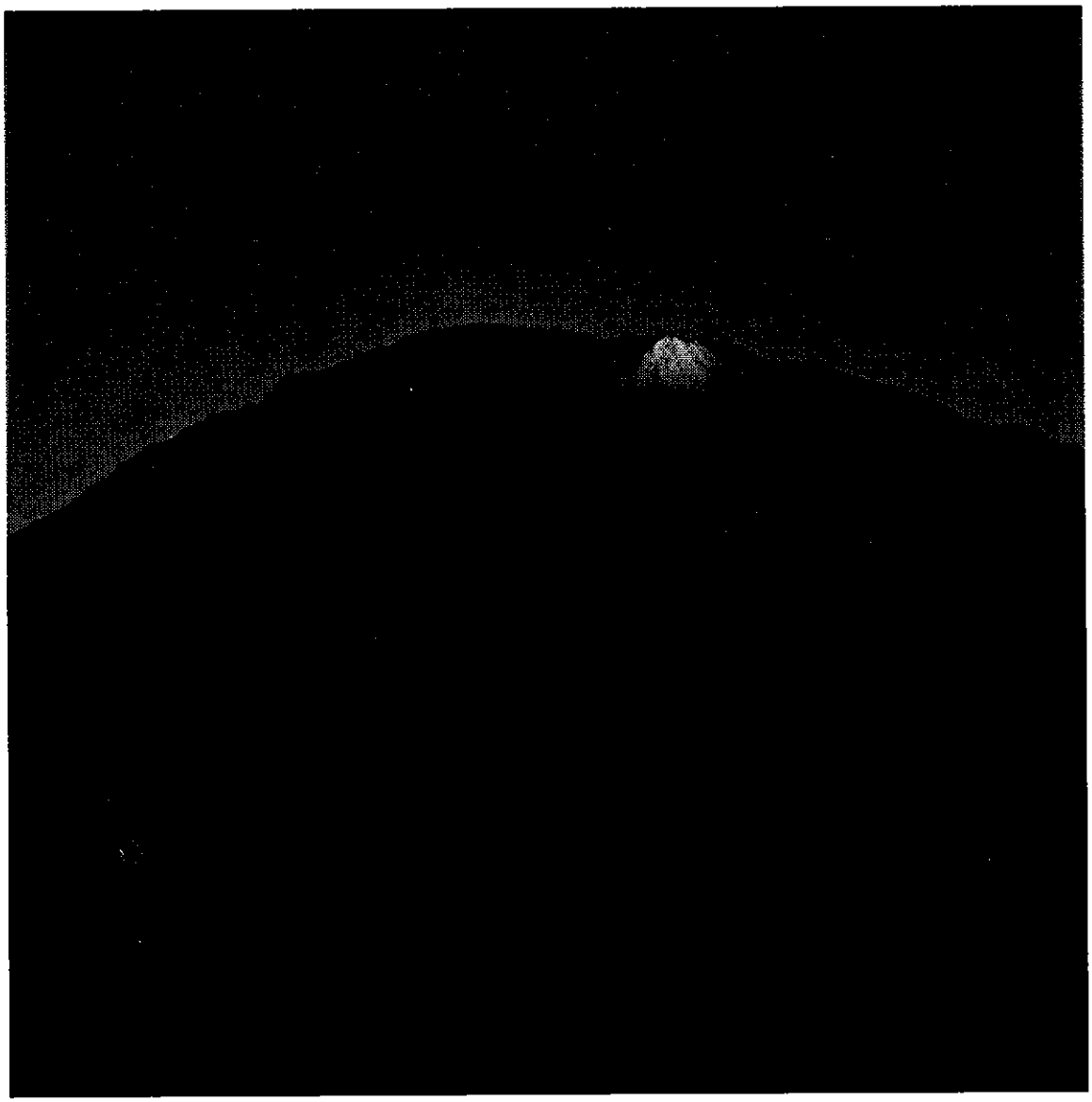


Figure 6: Fractal Model of Mt. Hood

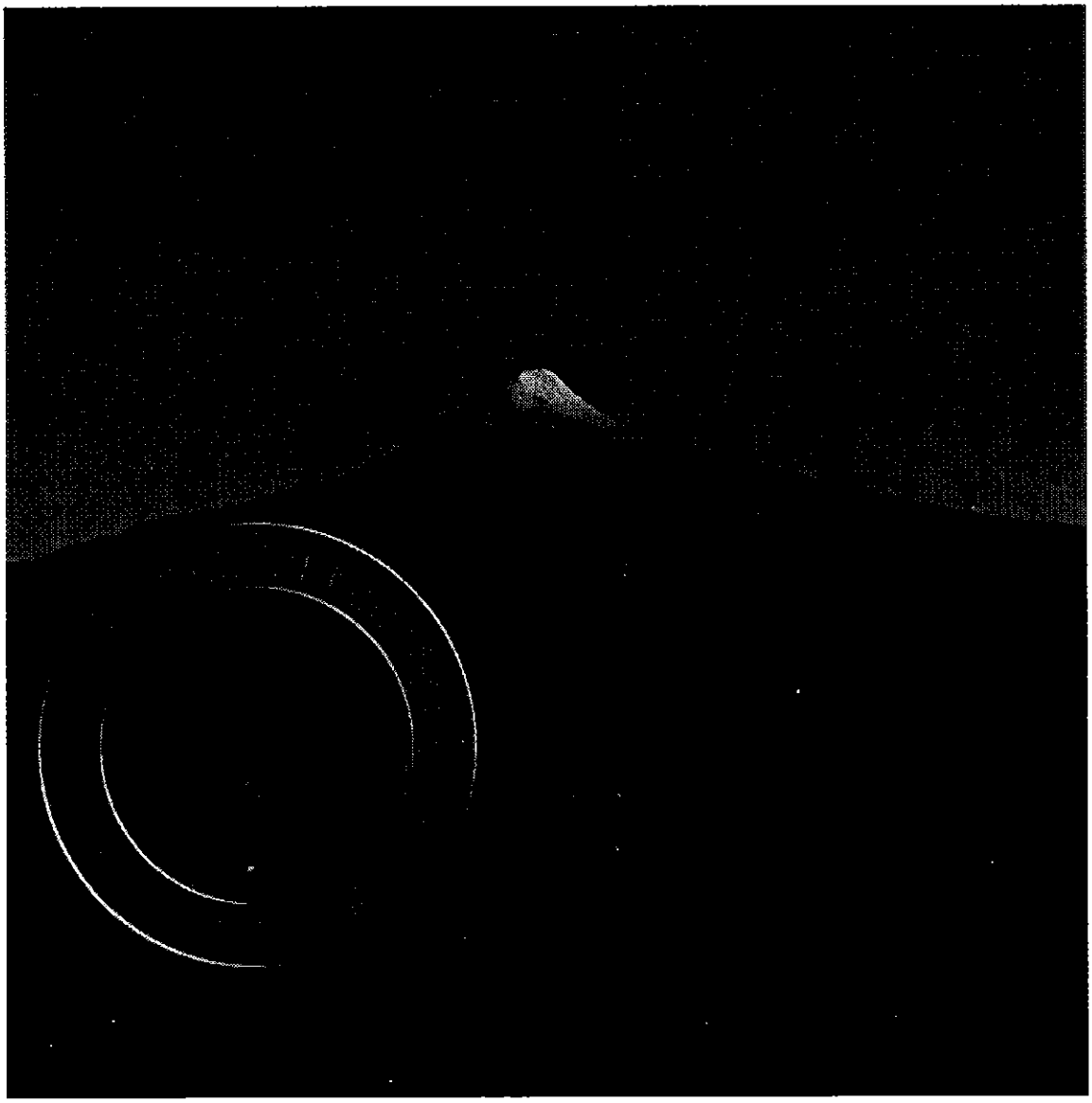


Figure 6 (cont): Fractal Model of Mt. Hood

has a relatively fast execution time. The asymptotic time complexity of the described midpoint algorithm, if applied to an n by n height field, is $O(n^2)$. Although this technique has several advantages, there are disadvantages that affect the quality of images produced. As has already been discussed, in the field of natural phenomena modeling, the midpoint displacement technique is infamous for producing artifacts. These artifacts manifest as a "crumpled paper," or a "creased" look in final images, producing results that are less than realistic [5]. Another disadvantage of the midpoint technique is that the geometric representation of the height field must be able to be subdivided in an equal and recurrent manner. The next model of fractional Brownian motion discussed in this thesis, the random faults technique, has the advantage that it can produce a rough fractal surface of general shape.

Random Faults

The random faults algorithm is very simple. Given a set of data that represents a general surface, the algorithm intersects the surface with a randomly generated plane, the site of the fault, and then offsets all of the points on one side of the plane by a pre- determined random number [9]. The following algorithm will calculate the effect of one fault on a set D of three-dimensional data points:

Generate a random plane P. P must, on the average, bisect the dataset D.
Calculate a Poisson random number R.
For each point d in D:

> If d is on the same side of the plane as the normal to the plane, offset d by distance R, in the direction normal to the tangent.

After a sufficient number of applications of the above algorithm, the result of any one fault disappears, and the data set represents a random fractal surface [9]. Intuitively, this technique works because of simple probability. If the intersection of the random plane with the dataset dictates that some particular point P be offset, then the chances are very high that close neighbors to P will also be on the same side of the plane, and thus also offset. The farther a point is from P, the less likely that it will be offset. This follows the Brownian motion paradigm; Brownian motion dictates that the elevation of any given point is similar to that of its neighbors, and the elevations of distant points may not be similar at all.

More formally, this model produces fBm with β held constant at 2, corresponding to a fractal dimension of 2.5. Brownian motion can be interpreted as "the cumulative displacement of a series of independent jumps" [9]. For instance, a fBm function $B(t)$ can be calculated by summing white noise:

$$B(t) = \sum_{i=1}^{t} W(i)$$

where $W(i)$ is a white noise function with Poisson distribution. This equation produces fBm with the mean square variation at frequency f proportional to $1/f^2$. The actual mean variation will be proportional to $1/f$. For example, if W produces values in the range $[0, 1]$ and $B(t)$ is evaluated for $t = 1 \ldots 16$, then for two points (t_1 and t_2) sampled at frequency $f = 2$, the expected value of $|B(t_1) - B(t_2)|$ is $(8)(1/2) = 4$. If t_1 and t_2 are sampled at frequency $f = 4$, then $E(|B(t_1) - B(t_2)|)$ is $(8)(1/4) = 2$. From this example, it should be clear that in relation to each other, the expectations of these points are scaled proportionally to $1/f$. In a similar fashion, the summation

of random faults on a sphere produces a fBm surface scaled proportionally to $1/f$.

Applying the random faults technique to a sphere is an effective method of simulating the topography of an entire planet. Figure 7 illustrates the results of increasing numbers of iterations of the random faults algorithm on a sphere. Plate 2 shows the use of this technique to synthesize an image of an earth-like planet and a moon. The planet was calculated using the random faults method, and a smooth blue sphere was overlapped over most of the rough terrain, giving the appearance of oceans. The atmosphere was the result of another application of the random faults method. For the clouds, the data points of a spherical height field were not used to determine terrain height, but were instead used to calculate the transparency or opacity of clouds.

As previously mentioned, the advantage of the random faults method is that it can be applied to an arbitrarily shaped surface. Usually, this technique is not used if other algorithms will suffice, because the random faults method is slow. If an n by n height field is processed with m random faults, then the time complexity of the algorithm is $O(n^2m)$. Well over n faults are generally needed for an n by n height field, so the time complexity will usually be worse than $O(n^3)$. In more concrete terms, the midpoint displacement algorithm generated the data for Plate 1 in about ten minutes. Using the random faults algorithm, the surface of the planet in Plate 2 was generated on the same computer in nine hours. This poor time complexity limits the usefulness of the random faults method.

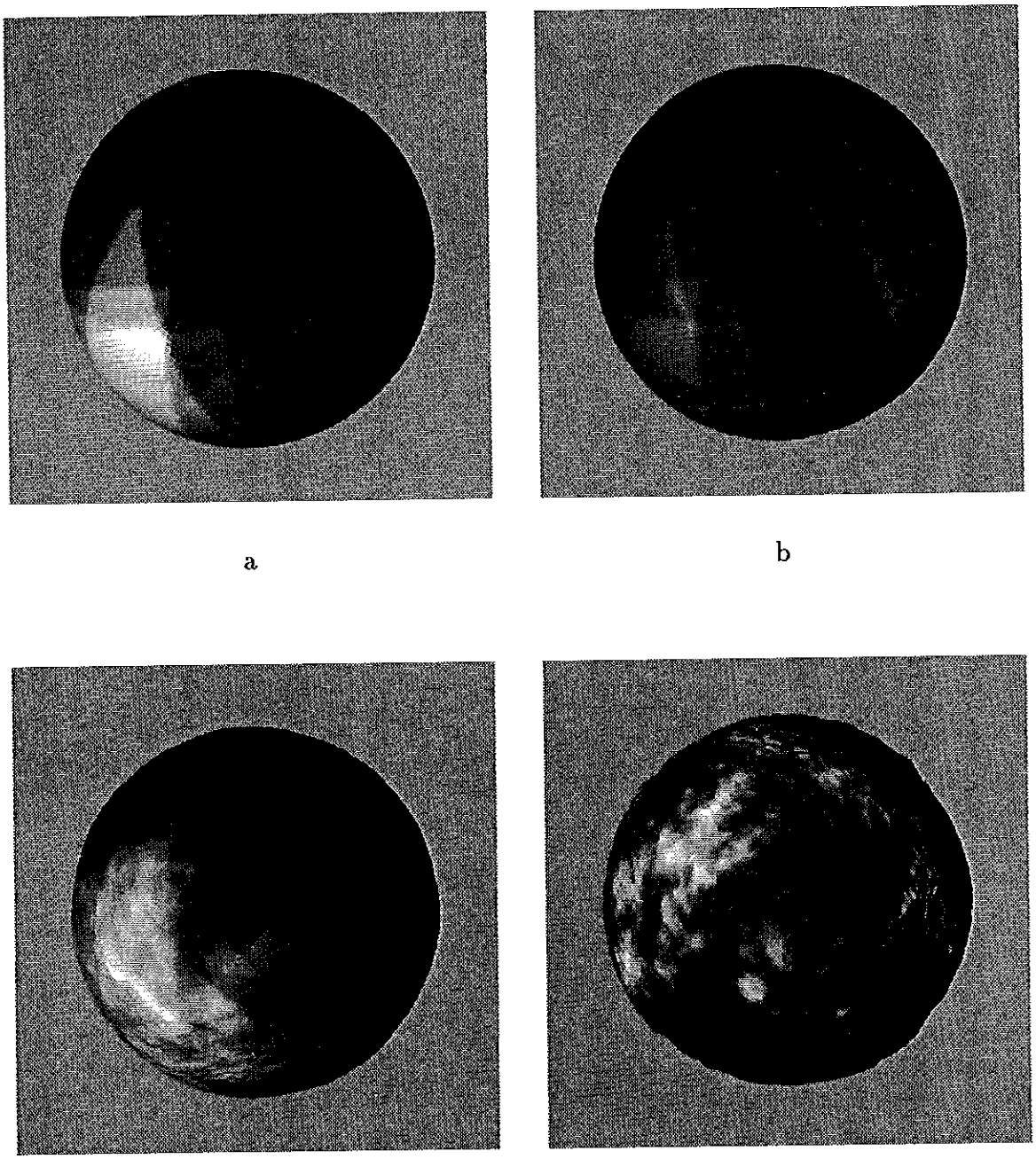


Figure 7: Stages of Random Faults Algorithm
These images illustrate the application of 16, 64, 256, and 1024 random faults.

Fourier Methods

Several methods of fractal terrain synthesis rely on the use of the Fourier transform. The Fourier transform is an important tool with applications ranging from natural sciences to signal processing; it has been referred to as "the prism of science" [10]. A prism separates white light into its spectrum of constituent colors; similarly, "the Fourier transform identifies and distinguishes the different frequency sinusoids (and their respective amplitudes)... [of] an arbitrary waveform", thus producing a spectral (frequency domain) representation of the wave [2]. Conversely, the inverse Fourier transform produces a time or spatial domain waveform from its spectral representation. Together, the Fourier transform and its inverse provide a means to synthesize or filter an arbitrary waveform based on its spectral representation.

Recall that the model of fractional Brownian motion dictates that the expected value of the mean square variation in amplitude at frequency f is proportional to $1/f^{\beta}$. Because this describes the relative scale of the frequency components of an fBm waveform, the Fourier transforms may be used to synthesize fractal terrain. A straightforward method of producing two-dimensional fBm involves rescaling the spectral representation of a random waveform. With this method, a two-dimensional height array is initially filled with random numbers. Applying the Fourier transform to this array will result in a spectral representation with completely uncorrelated amplitudes. A $1/f$-noise filter is applied to the spectral field, scaling values at frequency f by $1/f^{\beta/2}$. The application of the $1/f$- noise filter changes the random spectral representation to a spectral description of fractional Brownian motion. The inverse

Fourier transform is used to convert the data back into its spatial representation, thus producing a random fractal height field [6]. This filtering process involves the application of both the forward and inverse Fourier transforms. Saupe presents a more efficient method, spectral synthesis, which only requires one Fourier transform [9].

Spectral synthesis, as its name implies, involves the direct creation of an fBm frequency spectrum. An fBm spectral map is constructed by assigning, at each frequency f, a random amplitude with an expected value proportional to $1/f^{\beta/2}$. It is also important to randomly time-shift the spectrum, thus preventing symmetry in the peaks and valleys of the individual sinusoids. For a spectral representation H, an individual frequency f is shifted a constant t_0, producing H' by the following:

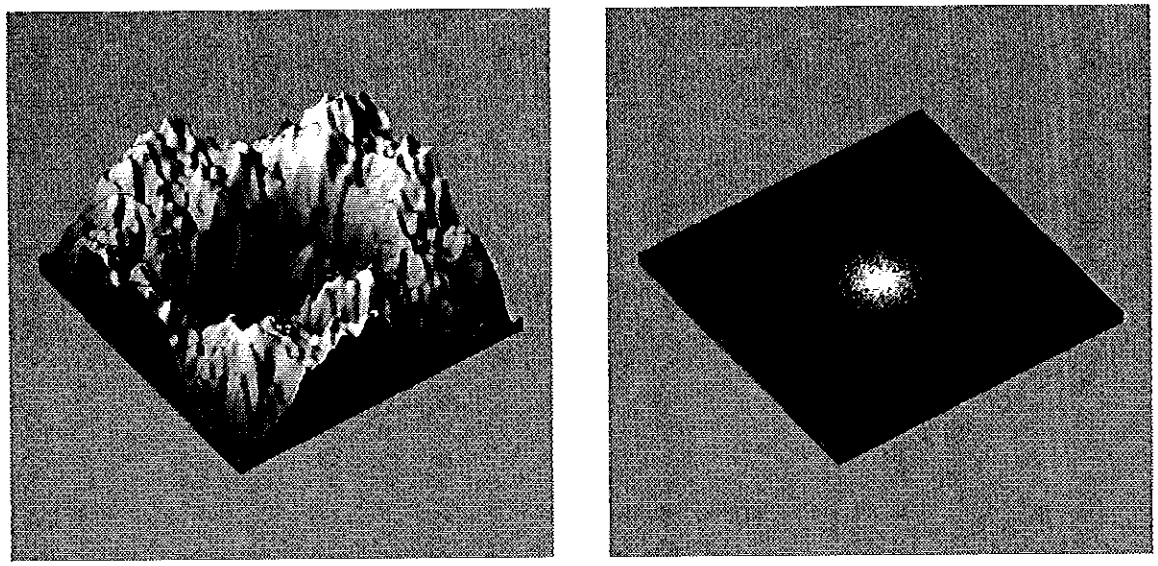
$$H'(f) = H(f)\,(\cos(2\pi f t_0) - i\sin(2\pi f t_0))$$

where $i = \sqrt{-1}$ [2]. This transform can be applied to all frequencies, with t_0 assigned at random, effectively eliminating symmetry of frequency components. Although time-shifting produces a complex spectral map, because the real part of the frequency domain is based on an even function (cosine) and the imaginary is odd (sine), applying the inverse Fourier transform will produce only a real spatial-domain height field [2]. As previously mentioned, this procedure eliminates the use of the forward Fourier transform. Synthesizing the height field can be done with a better asymptotic time complexity than a Fourier transform, thus improving the time constant of the entire process.

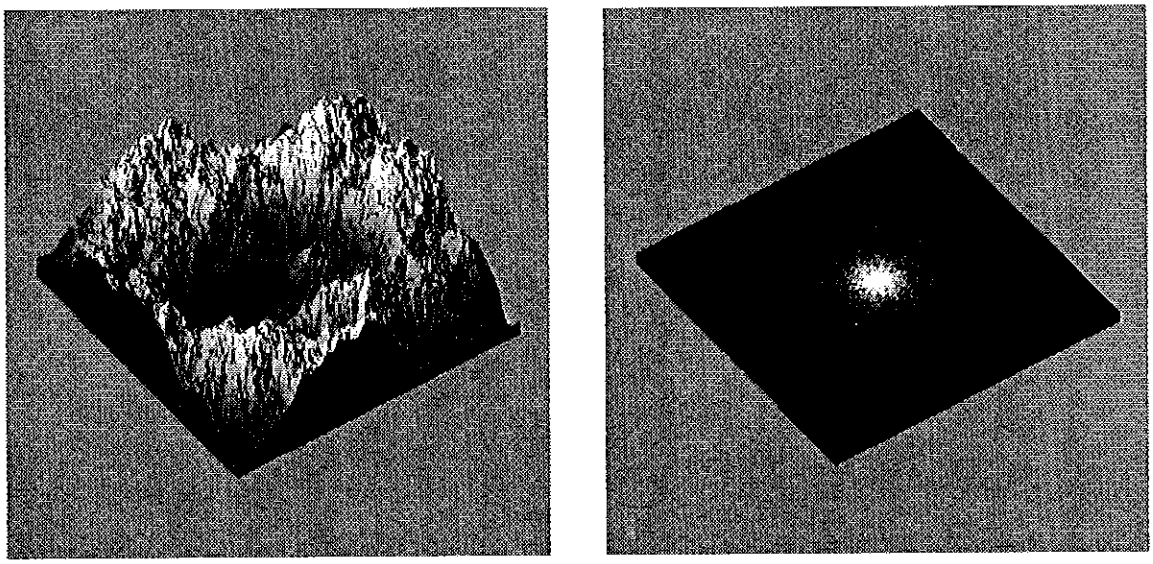
Figure 8d (left) is an example of a surface created using the Fourier spectral synthesis method. The spectral representation of this surface is shown to its right.

In the spectral map, the center of the height field corresponds to the amplitudes of low-frequency waveforms that constitute the fractal surface. Values at increasing radii from the center of the field correspond to amplitudes of increasing frequencies. Figures 8a through d illustrate the scaling of amplitude as frequency varies, and also help to show how the summation of sinusoids form a complex surface. Figure 8a displays the low-frequency components of the fractal surface in both spatial and spectral domains. Because of the $1/f$ scaling, we can expect the higher frequency sinusoids to be of smaller amplitude. This is evident in Figure 8b through d, which show the results of adding an increasing number of frequency components to the fractal surface.

Visually, the results of the Fourier techniques are better than both midpoint displacement and random faults. The Fourier methods are the most mathematically correct of the three models, but each is only an approximation to true Brownian motion. The time complexity of the Fourier method is dictated by the implementation of the Fourier transform. A fast Fourier transform is typically used, with a time complexity of $O(n^2 \log_2 n)$ for a two-dimensional n by n height field. A disadvantage of the Fourier techniques is that they must be applied to rectangular height field. Also, the initial height field cannot be seeded, resulting in lack of user control over the final landform distribution.



c



d

Figure 8 (cont): Example of Fourier Method

Conclusion

Fractal modeling is important in computer graphics, and many fractal models have worked successfully in the field of natural image synthesis. Models of fractional Brownian motion approximate the correlated and random detail of natural topography. In this thesis, three primary models of fBm were discussed: midpoint displacement, random faults, and Fourier techniques. For general purpose rendering of random mountains and rough fractal surfaces, the Fourier methods produce excellent, realistic results. The Fourier methods execute relatively fast, and are the most mathematically correct interpretations of fractal Brownian motion. If one needs control over the distribution or shape of the resulting height field, the midpoint displacement algorithm must be used. For instance, the midpoint displacement technique was used to produce a photo-realistic image of Mt. Hood, as seen in Plate 1. The random faults technique is extremely slow, but is simple to implement and is quite versatile. Using random faults, a fractal surface of any shape can be generated. If one wants, for example, to create a fractal surface in the shape of a torus, the random faults method is the primary option. Research and experimentation suggests that there is no fBm algorithm that will work adequately for all cases, but individual techniques are well suited for specific applications.

Future Work

Procedures that model fractional Brownian motion have primarily been used to synthesize natural topographic relief. Since their introduction, the basic random fractal techniques have been extended to simulate eroded terrain, clouds, and ocean waves. Although the fBm paradigm plays a significant role in the simulation of natural phenomena, it has received little attention from other areas of computer graphics. In particular, the random fractal model has the potential of providing a more accurate and natural means of simulating light scattering for surface illumination and shading. Many natural and man-made surfaces exhibit fractal roughness at a microscopic level. These microscopic surface characteristics affect the reflection of light and ultimately determine the surface's appearance. In theory, statistical surface roughness properties could be measured and used as control parameters for a model of fractional Brownian motion. Once the computer representation of the surface is generated, the interaction of light with the surface could be simulated, providing data to accurately shade an arbitrarily shaped object with those specific surface qualities. This example is quite removed from terrain synthesis, but as previously mentioned, many natural systems display the $1/f$-noise fBm model. Because this model is so prevalent, it seems that there are numerous random fractal applications that have yet to be discovered.

References

[1] Barnsley, Michael. *Fractals Everywhere.* San Diego: Academic Press, 1988.

[2] Brigham, E. Oran. *The Fast Fourier Transform.* New Jersey: Prentice-Hall, 1974.

[3] Fournier, Alain, Don Fussell and Loren Carpenter. "Computer Rendering of Stochastic Models." *Communications of the ACM* 25 (1982): 371-384.

[4] Mandelbrot, Benoit B. *The Fractal Geometry of Nature.* San Francisco: W.H. Freeman, 1983.

[5] Mandelbrot, Benoit B. "Technical Correspondence: Comment on Computer Rendering of Fractal Stochastic Models." *Communications of the ACM.* 25 (1982): 581-583.

[6] Mastin, Gary A., Peter A. Watterberg, and John F. Mareda. "Fourier Synthesis of Ocean Scenes." *IEEE Computer Graphics and Applications.* Mar. 1987: 16-23.

[7] Miller, Gavin S. P. "The Definition and Rendering of Terrain Maps". *Computer Graphics.* Aug. 1986: 39-47.

[8] Musgrave, F. Kenton. "Uses of Fractional Brownian Motion in Modelling Nature." *SIGGRAPH 1991 Course Notes: Fractal Modeling in 3D Computer Graphics and Imaging.* [Las Vegas]: ACM, [1991]. 5-34.

[9] Peitgen, Heinz-Otto, and Dietmar Saupe, ed. *The Science of Fractal Images.* New York: Springer-Verlag, 1988.

[10] Pickover, Clifford A. *Computers, Pattern, Chaos and Beauty.* New York: St. Martin's Press, 1990.

[11] Pokorny, Cornel K. and Curtis F. Gerald. *Computer Graphics: The Principles Behind the Art and Science.* Irvine: Franklin, Beedle and Associates, 1989.

[12] Sieradski, Allan J. "Fractals with Computers." Lecture. University of Oregon. Eugene, 20 Jun. 1991.