Graphics and Image Processing

James Foley*
Editor

Computer Rendering of Stochastic Models

Alain Fournier
University of Toronto
Don Fussell
The University of Texas at Austin
Loren Carpenter
Lucasfilm

A recurrent problem in generating realistic pictures by computers is to represent natural irregular objects and phenomena without undue time or space overhead. We develop a new and powerful solution to this computer graphics problem by modeling objects as sample paths of stochastic processes. Of particular interest are those stochastic processes which previously have been found to be useful models of the natural phenomena to be represented. One such model applicable to the representation of terrains, known as "fractional Brownian motion," has been developed by Mandelbrot.

The value of a new approach to object modeling in computer graphics depends largely on the efficiency of the techniques used to implement the model. We introduce a new algorithm that computes a realistic, visually satisfactory approximation to fractional Brownian motion in faster time than with exact calculations. A major advantage of this technique is that it allows us to compute the surface to arbitrary levels of details without increasing the database. Thus objects with complex appearances can be displayed from a very small database. The character of the surface can be controlled by merely modifying a few parameters. A similar change allows complex motion to be created inexpensively.

CR Categories and Subject Descriptors: I.3.3. [**Computer Graphics**]: Picture/Image Generation—*display algorithms*; I.3.5. [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid, and object representation*;I.3.7 [**Computer Graphics**]: Three Dimensional Graphics and Realism—*color, shading, shadowing, and texture.*

General Term: Algorithms

Additional Key Words and Phrases: fractals, terrain models, stochastic models

1. Introduction

Traditional modeling techniques used in computer graphics have been based on the assumption that objects are essentially a collection of smooth surfaces which can be mathematically described by deterministic functions. The simplest such technique assumes that objects are collections of polygons whose surfaces are obviously described by linear functions. Greater flexibility is achieved by the use of surfaces which are described by higher-order polynomials, as with Bezier [1] or B-spline surface patches [12].

These techniques have been quite successful in rendering realistic images of artificial objects, with their relatively simple macroscopic characteristics and their regularly periodic surface features. Natural objects, such as stones, clouds, trees, terrain, etc. are characterized in general by no such regular features or simple macroscopic structures, and these methods have been less effective in modeling them.

Macroscopic features of natural objects are often represented explicitly using large amounts of data. In the case of terrain, the information is usually obtained from contour maps, and in some fashion transformed into a surface represented by a large number of polygons [11]. Similarly, smoke has been modeled as volumes containing very large numbers of points distributed according to certain theoretical functions used in the study of smoke formation [8]. In both cases, capturing the macroscopic features to be modeled involves significant time and/or space requirements and the use of specialized techniques that are not generally applicable to other types of natural features. The problem is that these conceptually simple objects require a large number of

B. Mandelbrot, on whose work this paper is based, has raised certain objections which will be published in a subsequent issue.

This paper reports the results of two independent research efforts—one by Carpenter and the other by Fournier and Fussell. They both submitted papers to the 1980 SIGGRAPH conference, and through the conference to *CACM*. Both papers were accepted for *CACM* with the understanding that the authors would consolidate their work into a single integrated and definitive piece.—J. Foley.

* Former editor of Graphics and Image Processing. Robert Haralick is the current editor of this department, which has recently been renamed Image Processing and Computer Vision (see April '82 *Communications*, pp 311–312.)

Alain Fournier and Don Fussell's work was performed at The University of Texas at Dallas, and was partially supported by NSF Grant MCS-79-01168 and facilitated by the use of the Theory Net, NSF Grant MCS-78-01689. Loren Carpenter's work was performed while at Boeing Computer Services.

Authors' Present Addresses: A. Fournier, Computer Systems Research Group, 121 St Joseph St, University of Toronto, Toronto, Ontario M5S 1A1; D. Fussell, Department of Computer Science, The University of Texas at Austin, Austin, Texas 78712; L. Carpenter, Lucasfilm, P. O. Box 2009, San Rafael, California 94912.

modeling primitives (points, polygons, or patches) because they are visually quite complex. On the other hand, a conceptually or technologically complex object, like an airplane, can be very effectively modeled with a smaller number of such primitives.

Using a completely different approach, small-scale textures of natural objects have generally been modeled by some single repetitive texture function mapped onto all patches comprising such an object. However, the regularity of the effect detracts considerably from a natural appearance.

A fundamental limitation of these approaches is that objects are modeled at a predetermined, fixed scale regardless of its suitability for any particular viewing distance. Thus, from sufficiently far away, all but the most large-scale changes in terrain modeled by a fixed set of polygons may be invisible, rendering a large portion of the database and the processing required to display it superfluous. Likewise, a view of such terrain from very close up may reveal no more than a flat, featureless portion of a polygon, lacking any cues that it does indeed represent terrain. The latter problem may be alleviated somewhat by texture mapping, but with the usual static texture definitions it is still possible to get too close for the resolution of the texture pattern.

In many applications in which natural phenomena are to be represented, one is primarily interested in achieving sufficient realism in the representation of the objects for their nature to be easily recognizable. The specific features of any such objects on all but the most macroscopic scale are of secondary importance. For example, in a computer-generated animated sequence we may wish to have a mountain range which is obviously a mountain range but which is not intended to correspond to any particular real-world mountains. In such a case, we are interested only in the general size, shape, and position of the mountain range as specific features to be modeled explicitly. In order to make such an "object" recognizable as a mountain range, we would like to generate the macroscopic features that any typical mountain range would have. It would be advantageous to have a technique that would allow us to do this without the use of a large database to represent the object. In applications where one wishes to display real-world data, the addition of suitable information at various scales may be used to enhance realism. For example, in flight simulators, various types of terrain are represented by a few large polygons whose color, shape, and position provide vague cues as to their nature. If a pseudo-random rocky texture could be added to surfaces representing mountainous terrain, much more realistic images could be generated. The use of an extremely large, detailed database for such purposes would be prohibitive, while the use of traditional, deterministic texture mapping techniques would not be fully satisfactory.

The representation of motion in computer graphics systems has suffered, less obviously, from a similar limitation. Previous attempts to represent turbulent motion have been limited by the apparent complexity of the task. An effective means of generating an irregular surface with an irregular motion in a flexible way will allow the solution of such problems as realistically modeling a waterfall, rapids, or ocean waves, all of which present serious challenges to computer graphics researchers [19], [24].

All of the drawbacks mentioned above result primarily because most traditional models of real-world phenomena in computer graphics are totally deterministic in philosophy. There have been some exceptions, however. Early work by Mezei et al. [20] generated textures and irregular shapes by random techniques, and Blinn [2] improved the realism of previous shading methods by using a model based on probabilistic assumptions. Also, research in image analysis and pattern recognition has produced a body of results on the statistical analysis of texture as well as some interesting examples of image synthesis using stochastic techniques [10], [23], [21].

We propose to extend the flexibility of the mathematical modeling techniques in computer graphics by generalizing the assumptions made about the characteristics of an object's surface and of its motion. Our basic approach is to model both primitives and their motion as a combination of both deterministic and stochastic features. Thus the surface of an object may be a polynomial function of a set of predetermined locations, or it may be a stochastic function of those locations, or both. Likewise, the motion of an object may be described as a smooth function interpolating its initial and final positions, or it may vary irregularly along the way. In this paper, we introduce simple and efficient techniques for rendering a large class of stochastic models which can be used to represent a variety of natural phenomena.

2. Stochastic Models

In a traditional graphics system, the modeling system is the part where the objects are defined in terms of the basic building blocks: the modeling primitives. The modeling primitives mainly used have been points, lines, polygons, and parametric patches. We define here a new kind of modeling primitive.

A stochastic model of an object (or more generally of a phenomenon, to extend the concept of an object to include possibly a time parameter), is defined to be a model where the object is represented by a sample path (a realization) of some stochastic process of one of more variables.

Stochastic objects can be made from several stochastic modeling primitives just as traditional deterministic objects are built from, for example, polygons or parametric patches. Also, since the class of stochastic processes properly includes the deterministic functions, the definition of stochastic models includes all previously used primitives.

Table I. Possible Applications of Stochastic Models.

Dimension of Primitive	Dimension of Stochastic Process (number of parameters) One-D Process	Two-D Process	Three-D Process	Four-D Process
1	Intensity on a line, Intensity in time	Scalar field	Intensity in 3-D space	Intensity in 3-D space in time
2	Direction on a plane, Surface in time	2-D vector on a surface	Intensity and altitude on a surface	Intensity and altitude on a surface in time
3	Direction in space in time, Color in time	Normal to a surface, Color on a surface	Color in space, Vector field in 3-D space	Color in space in time, Moving vectors

At the level of resolution normally used, the natural objects to be modeled can be taken to be continuous and will need continuous stochastic processes to model them. Since ultimately the models will be used for display on discrete devices, it is very convenient to have a means of computing a discrete sample of the continuous model at the rate required by the resolution of the image. This would usually correspond to the Nyquist rate, but if antialiasing is needed the rate of sampling can be chosen to be higher.

It is now clear that the three elements required for stochastic modeling are: (1) an appropriate object (phenomenon) to be modeled; (2) a stochastic process to model it with; (3) an algorithm to compute the sample paths of this process.

Objects that have features with stochastic properties that are strong enough so that appreciable savings in both storage and processing are obtained by replacing the stored values for the stochastic features by the few parameters needed by the definition of the stochastic process are likely to be represented most effectively using stochastic models. To use signal processing terminology, an object which has a high noise/signal ratio is a good candidate. It should be noted, however, that the stochastic process might model what at first appears to be signal, as will be seen in the example given below.

The stochastic process to be used can have two kinds of origin.

—It can be a legitimate mathematical model of the phenomenon to be modeled. A model in computer graphics is not normally required to be a mathematical model, but, of course, it does not hurt if it is. The example given for terrain falls into this category.

—The stochastic process can be empirically chosen, with the parameters determined to fit a particular application. Techniques need to be developed which employ some sort of canonical stochastic processes, to be used in stochastic approximation the same way power functions, for example, are used in curve fitting.

Since the stochastic process used can be analytically defined, many traditional algorithmic techniques can be considered as means to compute the sample paths. One of the most effective for display purposes is the recursive subdivision technique, introduced by Catmull [5] for parametric patches, and most notably used by Clark [7] and Lane et al. [13]. The same technique can be used in the context of stochastic modeling, and the advantages are even more important here.

—The depth of the recursion will be controlled by the on-screen resolution, giving two important benefits. We never run out of details, since the process can always generate new data as we close in. We never produce more details than necessary, therefore the computational effort is always commensurate with the on-screen image complexity.

—The basic computational step in the recursive subdivision uses an interpolation formula. Interpolation formulas are in general much easier to compute than incremental ones, especially those for midpoint interpolation, therefore further lowering the computational cost.

Depending upon the phenomenon being modeled, the stochastic process will have dimensions from 1–4, and the computed sample path, or more exactly the stochastic element computed from the sample path will have dimensions from 1–3. The various possibilities are in Table I, and some of the applications are indicated. Since in addition the stochastic element can be composed with various deterministic modeling primitives, the development of a wide range of new modeling techniques will be required, and some new computational issues will be raised.

The following sections address these issues in the context of one particularly useful and interesting stochastic model.

3. Fractals: A Stochastic Terrain Model

Perhaps the most common natural phenomenon to be represented in current applications of computer graphics is terrain. Since terrain is generally characterized by randomly distributed features that are recognizable by their overall properties as opposed to specific macroscopic features (as in the case of the mountain range example), its strong stochastic properties make it a good choice for the application of a stochastic model.

As noted above, we require a stochastic process that is appropriate for modeling terrain and an algorithm for computing sample paths of the process. In the following section we describe a suitable process for modeling terrains as well as a variety of other natural phenomena. We will then proceed in the subsequent section to develop new techniques for rendering the sample paths and for the construction of stochastic primitives which are especially suited for use in computer graphics.

3.1 Fractional Brownian Motion

In 1968, Mandelbrot and van Ness introduced the term "fractional Brownian motion" (which will be abbreviated to fBm) to denote a family of one-dimensional Gaussian stochastic processes which provide useful models for many natural time series [14]. Since then, multidimensional extensions of fBm have been studied by Mandelbrot as models of a wide range of natural phenomena, including in particular terrains (in two dimensions) and the isosurfaces (positions in space at which some parameter has equal value) of turbulent fluids [16].

We give a brief description of fBm. Let u be a real parameter such that $-\infty < u < \infty$, and let w be the set of all values of a random function taken from a sample space **W**. Ordinary Brownian motion, $B(u, w)$ is a real random function with independent Gaussian increments such that $B(u + \Delta, w) - B(u, w)$ has mean zero and variance σ^2 and $B(u_2, w) - B(u_1, w)$ is independent of $B(u_4, w) - B(u_3, w)$ whenever the intervals (u_1, u_2) and (u_3, u_4) do not overlap. Let H be a real parameter such that $0 < H < 1$ and let b_0 be an arbitrary real number. The random function $B_H(u, w)$, called reduced fractional Brownian motion, is defined by

$$B_H(0, w) = b_0$$

$$B_H(u, w) - B_H(0, w) = [1/\Gamma(H + 0.5)] \left\{ \int_{-\infty}^{u} [(u - s)^{H-0.5} - (-s)^{H-0.5}]\, dB(s, w) + \int_{0}^{u} (u - s)^{H-0.5}\, dB(s, w) \right\}$$

Thus $B_H(u, w)$ is a moving average of $B(u, w)$ weighted by $(u - s)^{H-0.5}$. Note that $B_{0.5}(u, w) = B(u, w)$, so when $H = 0.5$ we obtain ordinary Brownian motion. Thus we have a family of random functions whose values at any value of u depend upon all past values of u.

As for ordinary Brownian motion, the increments of fBm are stationary. Typical sample paths for $H = 0.5$ (ordinary Brownian motion), $H = 0.3$, and $H = 0.7$ are given in Figs. 1, 2, and 3.

A Fourier analysis of samples of such functions shows no dominant frequency, but rather a range of frequencies at all orders of magnitude. Fractional Brownian motions are members of the class of "1:f noises" [14], that is, those signals in which the contribution of each frequency to the power spectrum is nearly inversely proportional to the frequency. Additionally, the increments of fBm are statistically self-similar. This means formally that $B_H(u + \Delta u, w) - B_H(u, w)$ and $h^{-H}[B_H(u + h\Delta u, w) - B_H(u, w)]$ have the same finite joint distribution functions. Intuitively these features of fBm indicate that we may observe a sample of one of these functions at any scale and perceive identical statistical features. A surface generated using fBm would thus possess macroscopic features up to the order of magnitude of the overall surface

Fig. 1. Ordinary Brownian Motion ($H = 0.5$).

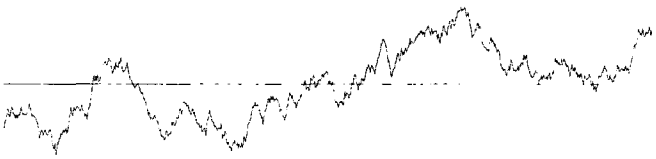


Fig. 2. Fractional Brownian Motion ($H = 0.3$).

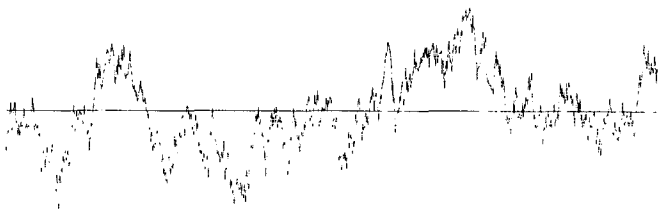
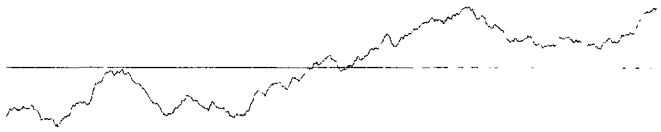


Fig. 3. Fractional Brownian Motion ($H = 0.7$).



generated, corresponding to the lowest possible frequencies in the Fourier spectrum of the sample, as well as arbitrarily small surface detail, corresponding to the higher frequencies in the Fourier spectrum.

3.2 Algorithms For Realizing Models Based On FBm

3.2.1 Algorithmic Requirements

In order for fractional Brownian motion to be generally useful for modeling in computer graphics, appropriate algorithms for computing its sample paths must be found. Since high quality images of complex scenes typically require that on the order of 10^6 sample points be generated, the efficiency of any such algorithm is obviously of critical importance. Not only should the asymptotic complexity of the algorithm be linear in the number of sample points generated, but the amount of computation involved in generating each sample point must also be as small as possible.

Although it is important, efficiency alone is not sufficient to make a sample path generating algorithm appropriate for use in graphics. In order to achieve the flexibility of deterministic models used in graphics, objects should be modeled piecewise as collections of stochastic primitives. Any modeling primitives in computer graphics must have two properties in order to be useful. The first of these, which we call *internal consistency*, is the reproducibility of the primitive at any position in an appropriate coordinate space and at any level of detail. That is, a modeling primitive should be rendered in such a way that its features do not depend on its position or orientation in space. In addition, the features visible when the primitive is rendered at high magnification should be consistent with those rendered at a coarser

scale. For deterministic primitives of any type, scale consistency is easily maintained on smooth curves or surfaces. Likewise, positional consistency (modulo the aliasing problem) is easy to maintain for primitives such as points, lines, or polygons, and for higher-order curves and surfaces has been achieved through the use of parametric definitions. Internal consistency of either type is, however, more difficult to maintain for stochastic sample paths and requires more care in the design of generating algorithms.

The other crucial property of modeling primitives is what we term *external consistency*. This refers to the continuity properties of adjacent modeling primitives. If modeling primitives are intended to share a common boundary, it must be possible to ensure that they are indeed continuous across this boundary at any scale at which they may be rendered. Additional consistency constraints such as derivative or higher-order surface continuity may be required in some cases, and other properties such as color may be subject to consistency constraints across primitives. As with internal consistency, this property has been easily maintained in the rendering of first-order primitives, although it has presented a serious research concern in the design of efficient algorithms for rendering higher-order deterministic curves and surfaces [7], [13]. Again, the problem of maintaining external consistency promises to be an even more serious concern in the design of algorithms for rendering stochastic primitives.

Let us again note here that when rendering any continuous analytically defined curve or surface, we are actually calculating a discrete set of sample points from the surface. These points are generally only approximations to the surface since even for deterministic functions the limited word size of a computer allows only for approximate representation of arbitrary real numbers. In computing sample paths of stochastic functions, it is often the case that only approximations can be calculated efficiently or at all, even leaving aside the numerical problems just mentioned. Nevertheless, such approximations are acceptable provided they are sufficiently good, which in computer graphics means that they meet visual criteria of indistinguishability from the actual sample paths. Indeed, since the process we are applying is a good model of terrain only on the basis of empirical statistical tests and not because they are derived from a theoretical model of terrain formation, any approximation which is sufficiently good to pass our visual test may itself be likely to be an equally good model by these statistical tests. In any case, visual acceptability as opposed to statistical criteria will be the basis on which we judge the quality of an approximation algorithm for graphical use.

3.2.2 Previous Algorithms

Mandelbrot has published a number of methods for calculating discrete approximations to fBm in various dimensions. These involve three basic approaches: a shear displacement process, a modified Markov process, and an inverse Fourier transformation.

The first uses the fact that fBm is the limit of a fractional Poisson field [17]. A fractional Poisson field in n-dimensions is a scalar field where at each point $\mathbf{P}$ the value of $F(\mathbf{P})$ is the sum of an infinite collection of steps (in the case of terrain, these steps can be seen as straight faults) whose directions, locations, and amplitudes are three sequences of mutually independent random variables. This method was used by Mandelbrot to generate the first computer simulation of a fractional Brownian surface. While it has solid theoretical foundations and has been used to produce striking pictures, it is not suitable for our applications, both for its $O(N^3)$ time complexity for surfaces, and for the fact that it is not clear that it could be adapted to our boundary constraints.

The second method is based on an algorithm to compute an approximation to discrete fractional Gaussian noise, which is the increment of fBm [15]. The algorithm computes what Mandelbrot called fast fractional Gaussian noise (ffGn) as a sum of a low frequency term and a high frequency term. The high frequency term is a Markov–Gauss process. The low frequency term is a weighted sum of M Markov–Gauss processes, M being a number proportional to $\log(N)$. The fast fractional Gaussian noise algorithm represents a considerable improvement in the computation of linear fBm, since its time complexity is $O(N\log(N))$ and its parameters can be adjusted to suit the observed time series if it is to be used in statistical analysis. Although some objection to the use of a two-dimensional extension to this method may be made on the grounds that its time complexity is greater than linear, a much more serious objection is that it appears that there is no valid extension of the method to two dimensions. Also, there seems to be no obvious method to adjust the computation to the needed resolution while maintaining any consistency.

The third approach, which also gives an $O(N\log(N))$ time complexity involves the generation of Gaussian white noise, in which all frequencies are equally represented, and then filtering it using fast Fourier transform techniques in order to force the different frequencies to fall off as required by the value of the parameter H for the particular fractional Gaussian noise desired. Fourier techniques were used by R. Voss to illustrate [18].

Each of the methods discussed above has its own theoretical and practical advantages. However, they have in common the drawbacks that their time complexity is greater than linear and that the basic operations involved in their computation are costly (involving transcendental functions). We will now present our own method for computing an approximation to fBm which avoids these drawbacks.

3.2.3 A Recursive Subdivision Algorithm

We have noted above the three basic requirements that an approximation algorithm appropriate for sto-

chastic modeling must meet, and we have discussed the advantages of the recursive subdivision algorithms for rendering models of any type in computer graphics. We now present such a recursive algorithm for generating approximations to the sample paths of one-dimensional fBm.

In order to be able to use this type of algorithm, the crucial requirement is that the distribution of the process for which samples are to be computed can be interpolated from the boundary points of the sample. Since one of the features of fBm is an infinite span of interdependence, it is not *a priori* obvious that such an approach would be successful. However, two facts help design an approximation algorithm.

—Fractional Brownian motion is self-similar. This means, as stated above, that the increments of $B_H(u)$ (for simplicity of notation, we will henceforth use $B_H(u)$ instead of $B_H(u, w)$) are such that $B_H(u + \Delta u) - B_H(u)$ and $B_H(u + h\Delta u) - B_H(u)$ have the same distribution if the latter is rescaled by a factor of h^{-H}, H being the self-similarity parameter.

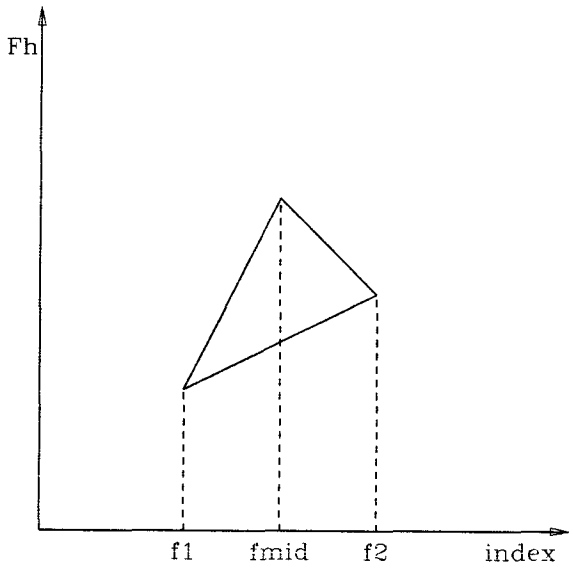
—A formula exists [14] for the conditional expectation of $B_H(u)$, $0 \le u \le 1$, knowing $B_H(0) = 0$ and $B_H(1) = 1$: $E[B_H(u) | B_H(1)] = \frac{1}{2}(u^{2H} + 1 - |u - 1|^{2H})$. When $u = \frac{1}{2}$, the right-hand side becomes $\frac{1}{2}$ independently of H.

These two properties give an estimate of the expected value and the variance of the increment of the process, which is all that is needed, since the process is Gaussian.

An algorithm designed using these properties is given below in Pascal. The function GAUSS(seed, index) returns a Gaussian random variable with zero mean and unit variance. It uses the variable "seed" as its seed. Explicit control over this seed is given in order to allow for external consistency as discussed below.

```
Declarations in main program:
  type result = array [0..maxsize] of real;
  var maxlevel, seed, i:integer; scale, h:real; Fh:result;
Procedure called:
  procedure fractal (maxlevel, seed:integer; h, scale:real);
  var first, last:integer;
      ratio, std:real;
  procedure subdivide (f1, f2:integer; std:real);
      var fmid:integer; stmid:real;
      begin
        fmid:= (f1 + f2) div 2;
        if (fmid <> f1) & (fmid<> f2) then
          begin
          Fh [fmid] := (Fh [f1] + Fh [f2])/2.0 + gauss (seed, fmid)*
            std;
          stdmid := std * ratio;
          subdivide (f1, fmid, stdmid);
          subdivide (fmid, f2, stdmid)
          end
      end; /* subdivide */
  begin
      first := 0;
      last := 2↑maxlevel;
      Fh [first] := gauss (seed, first) * scale;
      Fh [last] := gauss (seed, last) * scale;
      ratio := 2↑ − h;
      std := scale * ratio;
      subdivide (first, last, std)
  end; /* fractal */
```

Fig. 4. Computation of a Scalar Value by Subdivision.

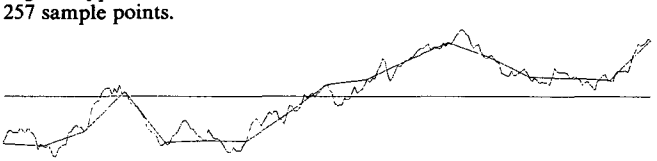


The algorithm recursively subdivides the interval [first, last] and generates a scalar value at the midpoint which is proportional to the current standard deviation times the scale or "roughness" factor (see Fig. 4). h is a parameter which determines the "fractal dimension" of the sequence output by the algorithms. (For a definition and discussion of fractal dimension, see [18].) It is equivalent to the H of fBm and can take on values between 0 and 1. Maxlevel determines the level of recursion needed. This algorithm is suitable for parametric applications since the recursion subdivides a parameter space into equal intervals. A similar algorithm which operates directly in a two-dimensional object space is given below. This algorithm is particularly suited for nonparametric subdivision.

```
procedure fractal(t1, t2, epsilon, h, scale:real; seed:integer);
  var f1, f2, ratio, std:real;
  procedure subdivide (f1, f2, t1, t2, std:real);
    var tmid, fmid:real;
    begin
      if (t2 − t1) > epsilon then
        begin
            tmid := (t1 + t2)/2.0;
            fmid := (f1 + f2)/2.0
                    + std*gauss(seed, tmid);
            std := std*ratio;
            subdivide (f1, fmid, t1, tmid, std);
            subdivide(fmid, f1, tmid, t1, std)
        end
      else output (f1, t1, f2, t2)
    end /* subdivide */
  begin
      f1 := gauss(seed, t1) scale;
      f2 := gauss(seed, t2) scale;
      ratio := 2↑ − h;
      std := scale*ratio;
      subdivide (f1, f2, t1, t2, std)
  end; /* fractal */
```

The sequence of scalar displacements generated gives an approximate sample path of one-dimensional fBm of parameter R. Unlike fBm, this approximation is neither stationary, isotropic, nor self-similar, as pointed out by B. Mandelbrot. This sample path can be used to create

Fig. 5. Typical Curve Obtained at Two Resolutions. h = 0.8, 17, and 257 sample points.



stochastic primitives as needed, as discussed in the following section. The graphs of Fig. 5 show typical samples at two resolutions for h = 0.6 and for 257 and 17 sample points. The two graphs are then from the same sample paths, but sampled by computation at different rates. This ability of the algorithm to generate discrete sample paths only at the rate needed makes it ideal for the purposes of stochastic modeling.

It is easy to see that the number of steps in the algorithm is a linear function of N, the number of sample points computed. Moreover, the amount of computation required to generate each sample point is small, requiring in the second case only 4 real additions, 1 subtraction, 3 real multiplications, and two divisions by 2 in addition to the generation of the pseudo-random variable. This makes it superior to the methods discussed above in terms of efficiency. By tying the random numbers generated at the endpoints to the values of t1 and t2, external consistency can be ensured since any adjacent sample paths generated with this algorithm would have the same endpoints. Internal consistency with respect to scale is assured by tying the seeds of the random number generator to the positions of the points calculated. Of course, internal consistency with respect to position is violated in this case unless t1 and t2 are assumed to be parametric variables and hence not subject to positional change. This can be avoided by using point-specific indices to compute the seed instead of the position, and using t1, t2 only for recursion control.

4. Applications of the Model

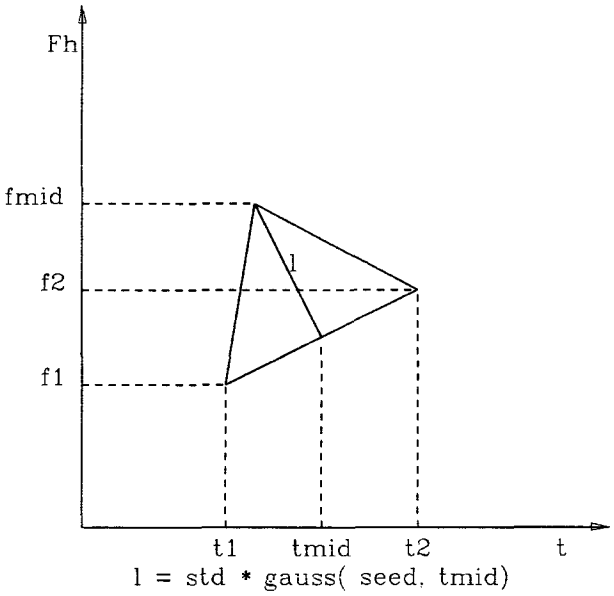
4.1 Creation of Stochastic Primitives

The most generally useful application of a stochastic model in graphics is in the construction of stochastic modeling primitives, which can be used for piecewise construction of objects with stochastic features. We describe in this section the construction of one and two-dimensional modeling primitives based on our recursive fBm sample path generator. We also discuss appropriate applications for these primitives and give examples.

4.1.1 One-Dimensional Primitives

The algorithm given in Sec. 3 for generating our approximations to fBm can be viewed as the construction of a "fractal polyline" primitive from an initial deterministic line segment. Of course, all displacements generated can either be viewed as offset vectors in the y direction of a two-dimensional coordinate system as

Fig. 6. Using the Scalar Value to Compute a Curve in the Plane.

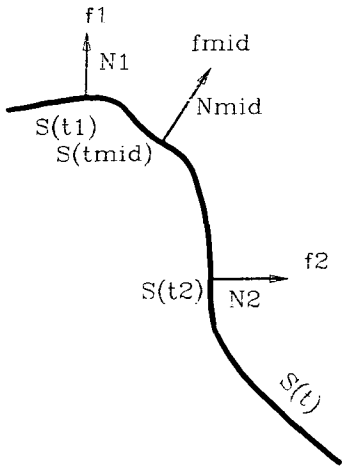


indicated in Fig. 4 or simply as scalar displacements as mentioned above. In the former case, rather unsatisfactory primitives are generated since displacements are tied to the coordinate system rather than the line segment from which the displacement occurs. To eliminate this coordinate system dependency, it is better to take the scalar displacement of the midpoint at each step in the recursion, and use it as an offset from that midpoint along a vector normal to the original line segment. This construction is illustrated in Fig. 6. The only inherent directionality in the resulting curve is that imparted by the slope of the original line segment at the highest level of detail. Figure 7 shows a typical curve resulting from such a procedure, with h = 0.5, with 2, 5, and 257 points.

In order to construct continuous curves from these fractal polylines the displacements of the endpoints of

Fig. 7. Typical Curve Obtained. h = 0.5, 0, 3, and 255 interpolated points.

Fig. 8. Construction for a Parametric Curve.



the initial line segment should be fixed at 0. This makes it trivial to guarantee the zero-order continuity of the curve produced. Higher orders of continuity of the fractal surface are meaningful only in a statistical sense since fBm has no derivative at any point [14]. It may be desirable to construct fractal curves based on smooth curves rather than the perimeters of polygons. In this case, the initial curve can be constructed piecewise, for instance, from either interpolating or approximating splines [12]. In this way, various statistical orders of continuity can be assured for this curve with derivative continuity being the most interesting. The scalar sequence generated by the subdivision process can be considered as displacements along vectors normal to the base curve at the appropriate midpoints in parameter space of the curve, as shown in Fig. 8. A more expensive alternative is to let the original spline curve be subdivided into two new spline curves with the original midpoint in parameter space becoming their common boundary and a new set of control points being generated. This common point is then displaced the generated random scalar distance along the common normal to the two curves at their boundary by displacing the adjacent end control points of the curves appropriately.

Any of the fractal polyline primitives constructed in these ways can be combined in arbitrary ways to construct representations of natural phenomena. For instance, the course of an imaginary river as it appears on a map could be generated using an appropriate value of h and level of scale. The instantaneous configuration of a bolt of lightning is also an appropriate candidate, as illustrated in the film *Vol Libre* [4]. An imaginary coastline on a map can also be created from fractal polylines like those of Fig. 7.

A more interesting application allows fractal primitives based on real data to be constructed using a technique we will call "stochastic interpolation." For instance, consider the polygon of Fig. 9 whose 8 vertices are sample points digitized from a map of Australia. The polygon is obtained as a linear interpolation of the positions of adjacent pairs of endpoints. However, it is

Fig. 9. Australia: 8 Sample Points.

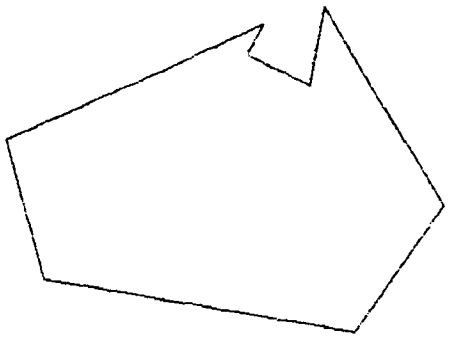


Fig. 10. Stochastic Interpolation. 8 original points and 8 × 127 interpolated points (h = 0.5).

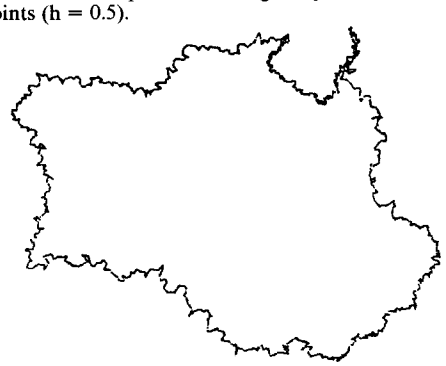
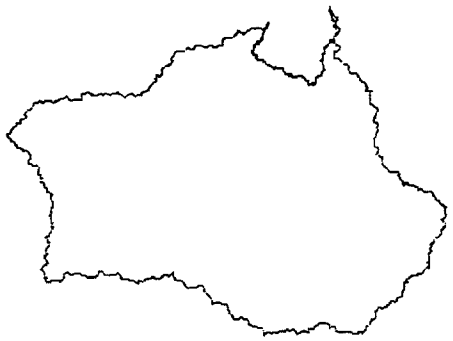


Fig. 11. Stochastic Interpolation. (h = 0.7).



well known that the coastline of Australia is very irregular when viewed at most any magnification, and so the regular polygon, although maybe recognizable as Australia by its overall shape, is not very realistic and looks nothing like the representation of the coastline presented on any reasonably accurate map. Moreover, empirical data suggests that the stochastic characteristics of Australia's coastline are nearly identical to those of one-dimensional fBm with $H = 0.87$ [18], [22]. Figures 10–13 show fractal polylines generated from the line segments of Fig. 9, with various values of h. All of them are much more realistic than Fig. 9, and Fig. 12 looks so real that those of us ignorant in geography would have difficulty arguing that this is not in fact the coastline of Australia traced from a map. Note that h in Fig. 12 is very close to the empirically measured value.

The visual evidence just cited provides a very strong argument that coastlines are best represented by curves

Fig. 12. Stochastic Interpolation. (h = 0.87).

Fig. 13. Stochastic Interpolation (h = 1.0).

with matching stochastic properties. All the real data obtained by digitizing the map is present in all of Figs. 9–13 since the endpoints of the line segments are not displaced in any case, but the stochastic interpolated curves give a much truer picture of Australia's coastline than a polygon does. In general, for natural phenomena with random, irregular characteristics, it can be argued that the quality of an interpolation between real sample points obtained from that phenomenon should be judged by the correspondence of its stochastic properties with those of the real sample itself.

4.1.2 Two-Dimensional Primitives

One of the most useful applications of a stochastic model in a three-dimensional environment is the representation of irregular surfaces, in this case, terrains. As in one-dimensional modeling, we wish to define a surface which is stochastic rather than deterministic, which at the same time maintains all the nice properties of the surface models currently most useful in computer graphics. We present two somewhat different approaches to the construction of two-dimensional fractal surface primitives. The first is based on a subdivision of polygons to create "fractal polygons" similar to the fractal polylines described above. The second is to define a stochastic parametric surface.

4.1.2.1 Polygon Subdivision. Consider a scene in which all surfaces consist of triangles. This type of model is very commonly used to represent real-world data which has been acquired automatically [11]. Each triangle can be subdivided into four smaller triangles by connecting the midpoints of the sides of the triangles. If the positions in three-space of these midpoints is obtained by a fractal polyline subdivision step given above, a single step in the rendering of a "fractal triangle" is obtained. These subdivisions can be continued until a level of scale is reached in which no triangle has a side exceeding a specified length. The original triangle is now a fractal triangle whose irregular surface consists of many small triangular facets.

A quadrilateral can be subdivided in a slightly more complex way. Generate the midpoint of each of the four sides using fractal polyline subdivision. For each of the two pairs of opposed midpoints, displace the midpoint of the line connecting them using the same procedure. The midpoint of the line connecting these two "midpoints" becomes the center point of the quadrilateral subdivision and four smaller quadrilaterals are generated. This process is continued as with triangles until the desired resolution is obtained, resulting in a fractal quadrilateral whose surface is composed of many quadrilateral facets.

If a scene is modeled by a mesh of triangles or quadrilaterals which are to be rendered as stochastic primitives using polygon subdivision, some care must be taken to ensure internal and external consistency. Internal consistency with respect to position requires that the seeds of the random number generator be indexed by some sort of invariant point identifiers rather than by functions dependent on the positions of the points. Internal consistency with respect to scale requires that the same random numbers be generated in the same order at each level of the subdivision, as before. External consistency is a bit trickier. Since adjacent polygons share a common boundary which must be subdivided, this subdivision must generate the same points on that boundary for both polygons. An obvious requirement is that the same random displacements must be generated on each boundary, which can be accomplished again by tying the seeds of the random number generator to identifers of points on the boundary, making certain that the same identifiers are assigned to the corresponding points in the representation of each polygon's boundary. However, if these displacements are allowed to be in a direction normal to the surface of the original polygon, problems arise when the adjacent polygons are not coplanar, as is generally the case. This is illustrated in Fig. 14. A solution is to calculate the normal of each point in the mesh as the average of the normals of the polygons containing it. Points randomly displaced along these normals will coincide when calculated for adjacent polygons, as desired. Of course, a similar problem exists for every new point calculated in the subdivision, even those completely internal to an original polygon. This can either be solved the same way, calculating the normals during the subdivision, or, less expensively, by letting all displacements be in a direction normal to the original polygon instead of averaging the normals of adjacent polygons created by subdivision.

Fig. 14. Gap Created by Tangent Discontinuity at the Boundary Between Two Polygons (0 original points, × interpolated points).

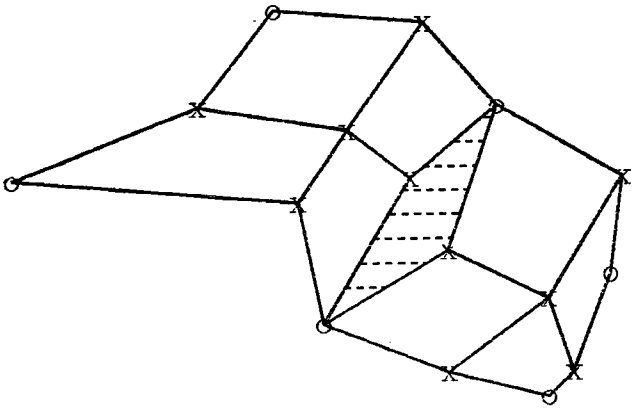
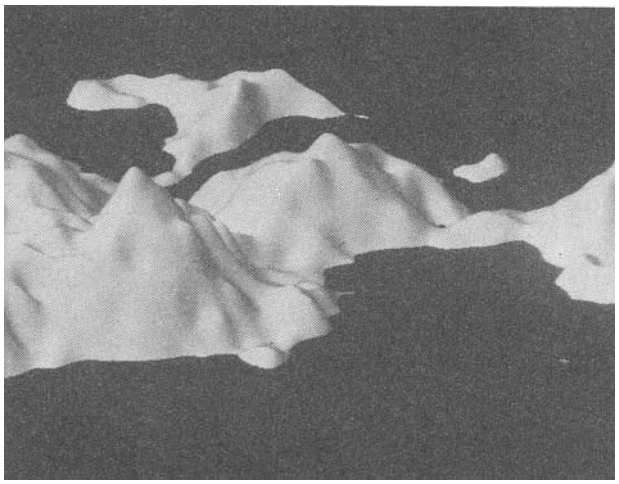


Fig. 15. Surface Produced Using the Stochastically Interpolated Points as Control Points for a B-spline Surface.



The primary advantage to this approach is the speed with which calculations can be done since only linear functions need be used. It does generate a surface which is self-similar within the range of scale covered by the subdivision and which does have a fractal dimension when carried to the limit. Thus its statistical properties are similar to those of two-dimensional fBm [14], although a better method for approximating fBm is given below. One important difference is that the surface generated in the limit is Markovian (for two-dimensional continuous processes, this means the values of opposite sides of an arbitrary boundary are independent given the boundary), while fBm, in which all sample points are correlated with all others, is not Markovian. As we have stated, however, our primary criterion is visual, and these methods can produce striking pictures of many terrains. The foreground of the cover picture, for instance, was produced using triangle subdivision. The most serious pitfall in using this method to produce good pictures is that derivative discontinuities across adjacent polygons can be annoyingly obvious in pictures that are not smooth shaded if the roughness factor used in the subdivision is not carefully chosen. (Note that smooth shading pictures of rugged terrain has a tendency to destroy the character of the surface.) The Markovian nature of the process, with no correlation between non-neighboring points, also tends to lead to the occasional generation of new polygons with radically divergent normals relative to other neighboring polygons during the subdivision process unless the random number generator is carefully constrained. Another way to obtain smooth surfaces is to use the computed stochastic points as control points of parametric patches, as was done to produce Fig. 15.

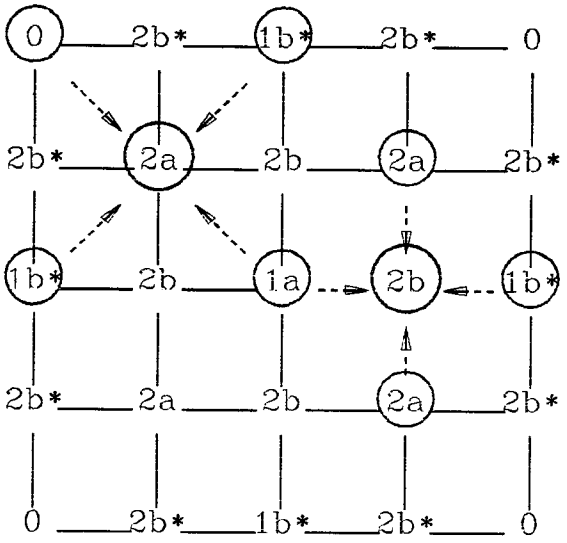
4.1.2.2 Stochastic Parametric Surfaces. Stochastic surface primitives can be created by extending deterministic parametric primitives as well as by polygon subdivision. In this case, we wish to define a surface description which is stochastic in nature rather than deterministic, which at the same time maintains the nice properties of the models currently most useful to represent complex objects in computer graphics. It is natural then, to consider functions of the form $\mathbf{X}(u, v) = \mathbf{P}(u, v) + \mathbf{R}(u, v, w)$, where $\mathbf{P}(u, v)$ is a vector-valued polynomial in u and v, and $\mathbf{R}(u, v, w)$ is a vector-valued random function on the sample space space $\mathbf{W}$, $w \varepsilon \mathbf{W}$. Thus $\mathbf{X}(u, v)$ is a two-dimensional stochastic process which we call *a stochastic surface function.* Intuitively, $\mathbf{P}(u, v)$ provides a way of defining the overall position of the surface while $\mathbf{R}(u, v, w)$ causes a stochastic variation in that position over the range of the parameters u and v.

$\mathbf{P}(u, v)$ can be any deterministic parametric function of two dimensions such as a bicubic or bilinear patch. $\mathbf{R}(u, v, w)$ is a vector normal to $\mathbf{P}(u, v)$ whose length is a random scalar $r(u, v, w)$. The calculation of $\mathbf{P}(u, v)$ and its normal are well-understood procedures for many surfaces which are useful in graphics [1], [5], [12]. We are interested in methods for generating $\mathbf{R}(u, v, w)$ as a two-dimensional extension of our fBm approximation algorithm.

The most straightforward approach is to use a method identical to the quadrilateral subdivision given above. This retains the drawbacks of that method, with the exception that normal averaging is unnecessary for those deterministic functions that assure derivative continuity across patch boundaries. If we compute the vector normal along with each subdivision, what is really needed is a non-Markovian approach which provides a better approximation to fBm across the surface of a patch. Of course, since we compute each patch separately, the overall surface cannot be strictly a fBm surface. If the parametric surface definition of the object has the proper stochastic properties globally, however, the approximation of the stochastic surface to fBm will be reasonable. An alternative would be to generate the entire stochastic surface at once, but this is impractical in most situations. Note that this difficulty, caused by the nonlocal character of fBm, does not arise in other stochastic processes of interest, making such computations easier.

To introduce the needed interdependence between points in the two-dimensional approximations to fBm, we will use the following scheme. First we compute the

Fig. 16. Order of Computation for Grid in Two Dimensions. (Order is 0, 1a, 1b, 2a, 2b, ···) * indicates points interpolated from boundary values only.



boundary of the patch, using the one-dimensional version of the algorithm to the level desired. We then fill the square for each level, computing the centers, then the sides, using at each step the four neighbors (diagonally for the centers, horizontally and vertically for the sides). At each step the new point is computed as a Gaussian pseudo-random variable, whose expected value is the mean of the four neighbors at this level, and whose standard deviation is $c^{-\ell H}$, with ℓ the level, H the self-similarity parameter, and c a constant to be adjusted to fit the application (see illustration in Fig. 16).

Figures 17–19 show a planet that has been generated with this technique using 10 bicubic Bezier patches. The "land" is made of patches with stochastic surfaces and the "sea" is made of the same patches with no stochastic component. The "coastlines" are then the zerosets of the two-dimensional fBm generated. Note that we used a depth-buffer algorithm to compute these intersections, but we could just as well have added the texture only

Fig. 18. Planets at Different World Space Resolutions but Similar Screen Space Resolutions.

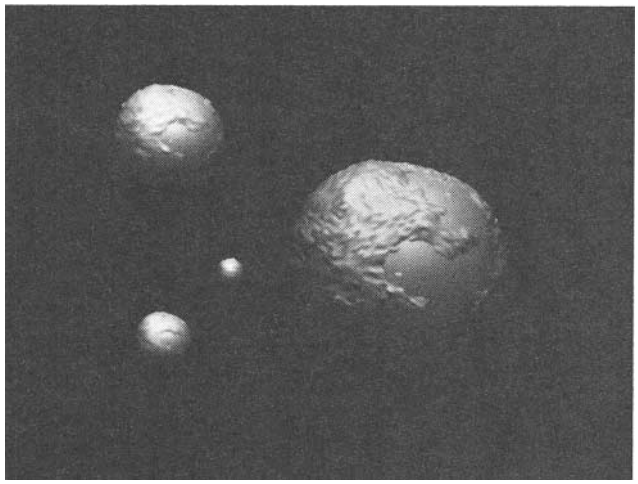
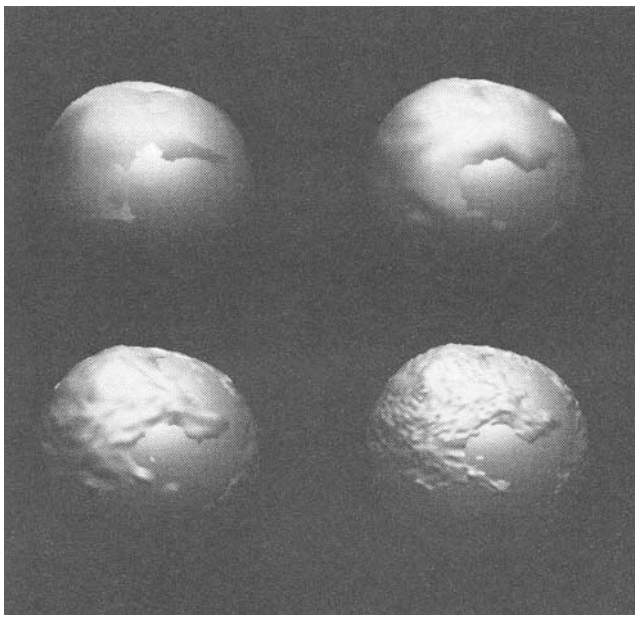


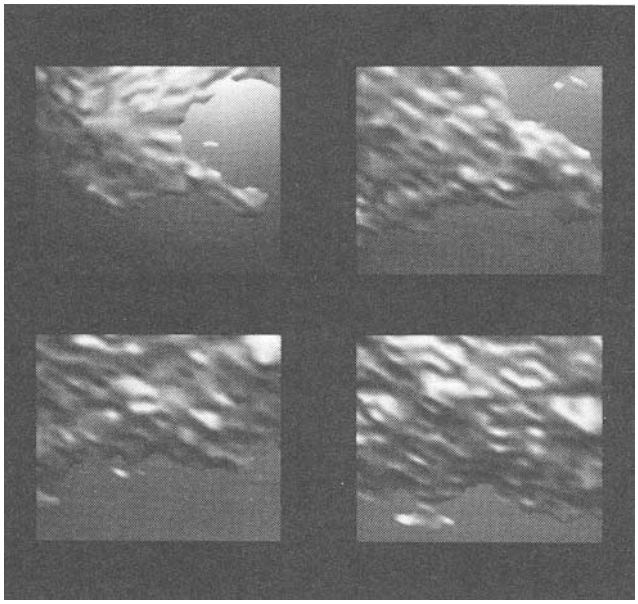
Fig. 17. Planets at Different Resolutions (coastlines are from depth-buffer computations).



where the displacement is positive and obtained the same "coastlines." The value of h has been chosen to be 0.6 since it is close to the empirical value obtained from actual measurements of geographic features [22]. The altitude has been exaggerated to give a more dramatic effect (the altitude of the highest peaks is about 10 percent of the radius of the planet). The subdivision has been stopped at a fairly low resolution, to illustrate the properties of the method, and the patches are actually processed as polygons (triangles to be specific) by the display system.

Figure 17 shows different resolutions for the planet at the same screen coordinate size, with the level of recursion being 2, 3, 4, and 5. At this on-screen size, though the overall appearance is similar, details, espe-

Fig. 19. Zooming in to the Planet.



cially for coastlines, are obviously different. This is due to the fact that the screen coordinate distance between computed points on the surface is much more than one pixel (about 20 pixels for the first planet of Fig. 17), and at the next level of computation the midpoint can be above or below sea level, changing locally the appearance of the coast. Features whose size is about the distance between points (in screen coordinates) can be significantly altered. For this reason in normal practice this distance should be kept at or below the size of a pixel.

Figure 18 shows the four objects together at sizes such that the on-screen resolution is the same. Observe that even though we are still above the pixel level (the average distance between computed points is about 2 pixels), the quality of the picture is satisfactory, and there are no noticeable differences in the appearance of the four planets. Considering that the whole database for the planet consists of only 90 three-dimensional points (defining the 10 Bezier patches), the comparison with a picture produced from a real terrain database which could require several 10,000 triangles for a comparable visual complexity is highly favorable. Of course, if the reproduction of a specific set of surface features obtained from cartographic data, for example, were required, it would still be necessary to model these features deterministically to the level of detail desired, and the database would grow accordingly.

Figure 19 illustrates how the process can be continued to zoom in to the surface to any desired level of detail, while keeping the same on-screen resolution. On the upper left is an easily recognizable part (the lower left corner) of the planet. On each of the other views, the central area is enlarged about twice. The main features of each view carry over to the next one, while new details appear. Here the average distance between computed points is about 6 pixels. This process can be continued further, still *with no modification* of the database, until we are arbitrarily close to the surface. Care has to be taken, however, because as the differences between two neighboring points become very small, the computation of the surface normals and the comparisons of the depth values in the Z-buffer can become inaccurate. The zooming in and out process can be repeated as often as desired since the particular stochastic surface generated is fixed and reproducible.

Parametric techniques will generally require somewhat more computation than polygon subdivision since the nonlinear deterministic functions involved require more computation for the rendering of points. In addition, since the recursive subdivision is done in parameter space, it is difficult to tie the depth of the recursion to the final distances apart in world or screen coordinates of the sample points generated. On the other hand, most of the difficulties cited for polygon subdivision are solved using this method. In particular, a surface is generated which has non-Markovian properties very close to those of fBm, and thus provides a much closer approximation. As a result, the value of h in the subdivision corresponds closely to H of fBm, so that empirical determinations of this value can be directly employed to generate terrain representations with characteristics similar to the measured surface, alleviating much experimental "twiddling" of parameters. Also, the higher correlations between points on the patch eliminate the need for tight control of the random number generator to avoid the glitches mentioned above.

The cost of the computation of the surface is a linear function of the number of points *displayed*. The cost of computation of the stochastic variables can be lowered using table lookup techniques (note that the numbers used do not need to pass very stringent tests for randomness). This indicates that the increase in computational cost will be small relative to the cost of the usual transformations and shading algorithms.

These algorithms share the general advantages of subdivision algorithms. They allow continuing the computation of the texture down to the pixel level, or even the subpixel level if some anti-aliasing is needed, while at the same time keeping the level of surface details constant as the object gets larger or smaller in screen space.

At the other end of the range in screen space, if the object is much larger than the screen size, the texture should be computed to the highest level of detail only for the portion of the patch or polygon that is not clipped out. Since such a subpatch or subpolygon cannot be computed solely on the basis of local information, some points outside of the displayed area are needed. It can be shown [9] that the total number of sample points to be computed is bounded by a linear function of the number of points to be displayed. So this algorithm allows "zooming" in and out on the surface, keeping the same displayed level of complexity (within one binary order of magnitude), while the time and space complexity grows only linearly as a function of the number of points actually displayed. This is then an implementation of a truly hierarchical approach to surface modeling, the importance of which was pointed out by Clark [6].

Another interesting feature of the algorithm for practical applications is that it is easy to change the value of the parameter h at any level of the computation. Therefore a terrain that looks very rugged from a distance (a low value of h), can become rather smooth at a higher scale (a high value of h). This models what happens if valleys are filled with sediments, for instance. This is a particular example of a general technique, namely changing the characteristic of the stochastic process, or even the stochastic process itself, according to the recursion level.

In our planet example, the nonstochastic components of the stochastic surface are the patches defining a close approximation of the sphere. As a result, the macroscopic features of the land masses are not predetermined. In most applications, however, the macroscopic features would be known, and some points of the surface would have the actual measured coordinates. In this case, it is

better (and easy) to force the stochastic component to be zero at these points. Thus the stochastic surface will interpolate these points, and we have a method for stochastic interpolation in two dimensions. Of course, the polygon subdivision methods generate no displacements at the original vertices and thus always produce stochastic interpolations of these vertices.

5. Further Applications of the Model

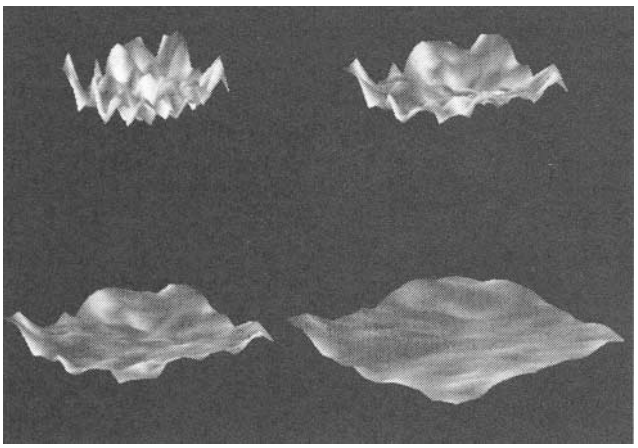
5.1 Other Stochastic Surface Properties

We have thus far only considered the application of fBm and other stochastic models to the creation of primitives whose surface position has stochastic characteristics. Other properties of a surface, such as its color, might also be allowed to vary stochastically. For instance, another instance of two-dimensional fBm with a high value of H and a low roughness factor could be used to determine the color of the surface of the planet. Of course, this property should also be continuous across patch boundaries. Another technique for color variation which can be used with polygon subdivision requires that a color be initially assigned to each vertex of the polygon to be subdivided. When a midpoint is computed for a side of the polygon, its color becomes that of one of the endpoints of the side. Which endpoint is chosen is decided according to a Boolean random function. When the subdivision is complete, the color of each facet's surface can simply be taken as the average of the colors of its vertices. This technique was used in generating the color variations and snow cap on the mountain range in the foreground of the cover picture.

5.2 Motion

Although various effective techniques have been developed for creating a series of images of a scene in which smooth, continuous motions of objects in the scene are depicted, these tend not to be very effective in handling complex irregular motions such as the path of a lightning bolt or the motion of a leaf in the wind. Stochastic techniques can provide powerful means of modeling motion which would have been difficult or impossible to represent otherwise. Consider, for instance, the action of unfolding a crumpled piece of paper. Figure 20 is four frames from a sequence representing such an action. These frames were generated using Bezier patches mapped with approximations to fBm with varying values of h. As h is changed from 0.3 to 0.9, the patch is rescaled to keep its surface area constant.[1] Thus a complex motion that would have been very expensive to generate previously is modeled very easily with stochastic techniques. Another example is the motion of a simple lightning bolt. The path of the lightning bolt can be represented as a fBm function from one dimension into three, like Brownian motion of a molecule in three-space. By simply changing the random numbers generated, while keeping the endpoint displacements fixed at 0, a sequence of instantaneous positions of the lightning bolt is created. Generating the same number of sample points in each instance, and allowing the motion of each sample point to interpolate the positions of that point in each of the "key frames" generated above, the motion of the lightning can be generated. Note that the interpolated path of each sample point can be created using either a deterministic or a stochastic technique. The lightning in the film *Vol Libre* [4] was generated in this way.

Fig. 20. Motion by Variation of h(h = 0.3, 0.5, 0.7, 0.9).



6. Conclusion

We suggest that recognition of the importance of the stochastic properties of the real world will lead to greatly increased flexibility in the modeling techniques used in computer graphics, just as probabilistic models have contributed significantly to the development of several related disciplines. We have applied Mandelbrot's fBm model for terrain and other natural phenomena and have developed efficient and appropriate sample path generating algorithms. We have shown several methods for creating stochastic modeling primitives of one and two-dimensions based on these algorithms and have demonstrated the use of stochastic interpolation of real sampled data points to create realistic representations of sampled phenomena. These methods constitute very natural and compact hierarchical object descriptions which are applicable to the modeling of various natural phenomena at a small fraction of the cost of deterministic methods of comparable quality, when these exist at all.

The techniques presented here barely scratch the surface of the possibilities of the stochastic approach to modeling. The most immediate extensions of this work are to use the same techniques to modify surface char-

[1] Note that a real fBm surface has infinite area, although our discrete approximations to it are, of course, finite. See [18].

acteristics other than position, for example, to create stochastic color patterns as has subsequently been done in the movie *Peak* by Mark Snilily, or to render small scale texture by stochastic variation of surface normals analogous to Blinn's method [3]. In contrast to these one and two-dimensional stochastic methods, the study of three and four-dimensional stochastic models should lead to interesting techniques for the representation of complex volumes and motions.

As indicated above, there are two general sources of stochastic models that may be of use in graphics. Although in this paper we have illustrated a mathematical model useful in representing terrain, there might be many natural objects for which it is unlikely that one will find a suitable mathematical model. Techniques which allow the empirical determination of parameters of a flexible canonical stochastic model which fit specific natural objects would be very useful in this regard. Research in the development of such techniques holds the promise of rich rewards for computer graphics.

Acknowledgments. The first two authors would like to thank Zvi Kedem for his many helpful suggestions and overall support, and Henry Fuchs, who taught them how to make pictures with computers. We thank Benoit Mandelbrot for providing inspiration through his book, and for his kindness and encouragement. We also thank Martin Tuori and Martin Taylor of DCIEM in Toronto, who helped in producing Figures 17 to 20.

Received 3/80; revised 12/81; accepted 2/82.

References

1. Bezier, P. Mathematical and practical possibilities of UNISURF. In Barnhill, R.E. and Riesenfeld, R.F. (Eds.). *Computer Aided Geometric Design*, Academic, (1974).
2. Blinn, J.F. Models of light reflection for computer synthesized pictures. In *Proceedings of SIGGRAPH '77.* Also published as *Comput. Graphics, 11*, 2, (Aug. 1977), 192–198.
3. Blinn, J.F. Simulation of wrinkled surfaces. In *Proceedings of SIGGRAPH '77.* Also published as *Comput. Graphics, 12*, 3, (Aug. 1978), 286–292.
4. Carpenter, L.C. *Vol Libre.* Computer generated animated movie. First Showing at SIGGRAPH '80 (July 1980).
5. Catmull, E. Computer display of curved surfaces. In *Proc. IEEE Conference on Computer Graphics, Pattern Recognition and Data Structure.* (May 1975).
6. Clark, J.H. Hierarchical geometric models for visible surface algorithms. *Comm. ACM, 19*, 10, (Oct. 1976), 547–554.
7. Clark, J.H. A fast algorithm for rendering parametric surfaces. In *Proceedings of SIGGRAPH '79.* Also published as *Computer Graphics, 13*, 2 (Aug. 1979), 174.
8. Csuri, C., Hackathorn, R., Parent, R., Carlson, W., and Howard, M. Toward an interactive high visual complexity animation system. In *Proceedings of SIGGRAPH '79.* Also published as *Comput. Graphics, 13*, 2, (Aug. 1979), 289–299.
9. Fournier, A. *Stochastic Modeling in Computer Graphics.* Ph.D. Dissertation, University of Texas at Dallas, (1980).
10. Fu, K.S. Syntactic image modeling using stochastic tree grammars. *Computer Graphics and Image Processing, 12*, (1980), 136–152.
11. Fuchs, H., Kedem, Z.M., and Uselton, S.P. Optimal surface reconstruction from planar contours. *Comm. ACM, 20*, 10, (Oct. 1977), 693–702.
12. Gordon, W.J. and Riesenfeld, R.F. B-spline curves and surfaces. In Barnhill, R.E. and Riesenfeld, R.F. (Eds.), *Computer Aided Geometric Design*, Academic, (1974).
13. Lane, J.M., Carpenter, L.C., Whitted, T., and Blinn, J. Scan-line methods for displaying parametrically defined surfaces. *Comm. ACM, 23*, 1, (Jan. 1980), 23–34.
14. Mandelbrot, B.B. and Van Ness, J.W. Fractional Brownian motions, fractional noises and applications. *SIAM Review, 10*, 4, (Oct. 1968), 422–437.
15. Mandelbrot, B.B.. A fast fractional Gaussian noise generator. *Water Resources Research, 7*, 3, (June 1971), 543–553.
16. Mandelbrot, B.B. On the geometry of homogeneous turbulence, with stress on the fractal dimension of iso-surfaces of scalars. *J. Fluid Mechanics, 72*, 2, (1975), 401–416.
17. Mandelbrot, B.B. Stochastic models for the earth's relief, the shape and fractal dimension of coastlines, and the number area rule for islands. *Proc. Nat. Acad. Sci. USA, 72*, 10, (Oct. 1975), 2825–2828.
18. Mandelbrot, B.B. *Fractals: Form, Chance and Dimension.* Freeman, San Francisco, (1977).
19. Max, N. Vectorized procedural models for natural terrains: Waves and islands in the sunset. In *Proceedings of SIGGRAPH '81.* Also published as *Comput. Graphics, 15*, 3, (Aug. 1981), 317–324.
20. Mezei, L., Puzin, M., and Conroy, P. Simulation of patterns of nature by computer graphics. *Information Processing* 74, 52–56.
21. Modestino, J.W., Fries, R.W., and Vickers, A.L. Stochastic image models generated by random tessellations in the plane. *Computer Graphics and Image Processing, 12*, (1980), 74–98.
22. Richardson, L.F. The problem of statistics of deadly quarrels. *General Systems Yearbook, 6*, (1961), 139–187.
23. Schachter, B. and Ahuja, N. Random pattern generation process. *Computer Graphics and Image Processing, 10*, (1979), 95–114.
24. Schachter, B. Long crested wave models. *Computer Graphics and Image Processing, 12*, (1980), 187–201.