The Definition and Rendering of Terrain Maps

Gavin S. P. Miller

Cambridge University Engineering Department

Cambridge, England.

Abstract

This paper examines three methods, two existing and one new, for the generation of fractals based on recursive subdivision. Both existing methods are found to have defects, which are not present in the new method. A parallel processing algorithm is proposed for the rendering of height fields which is exact and distributes the load evenly between the processors. A method is described for the 'fan-tracing' of height fields to allow the realistic simulation of water reflections.

Key words and phrases: computer graphics, terrain, height fields, fractals, scanline algorithms, parallel algorithms.

1. Introduction

The rendering of terrain models using computer graphics is an important problem because of its relevance to flight simulation, animation and CADCAM, to mention but a few applications. Special problems are posed by the realistic rendering of landscape because of the amount of detail required. Fractal methods have been proposed to allow database amplification, which is the generation of controlled random detail from a fairly sparse description [9].

An alternative approach is to use texture mapping methods on a few simple primitives. The texturing is defined procedurally, which means that the textured elements may be expanded without loss of high frequency detail, or shrunk without the occurrence of aliasing artifacts. Such texturing helps to blend together the crude approximate surfaces [12].

However problems remain. Fractal subdivision methods are slow and generate defects due to what is known as the 'creasing problem' which is the occurrence of creases or slope discontinuities along boundaries. Texture map methods, on the other hand, display visible discontinuities in texture gradient where two surfaces intersect. Also the outlines are smooth rather than irregular.

True parallax cues are important in generating a sense of movement and visual realism. Unfortunately, the texture mapping of a few simple primitives does not give correct local height variations. A typical state-of-the-art flight simulator can render about 5000 polygons in real time whereas a detailed terrain may consist of one quarter of a million elements. To achieve this degree of detail in real time will require a careful partitioning of the computational load between hundreds of processors working in parallel. Fortunately the special geometrical properties of height fields allow this.

2. A New Fractal Method

Recursive subdivision methods are preferable to Fourier transform methods because they are linear in time with the number of elements rather than NlogN. Recursion also allows the computation of a surface to varying degrees of detail in different regions based on the current projection. The two methods current in the literature are triangular edge subdivision, and what for this paper will be called diamond-square subdivision [9].

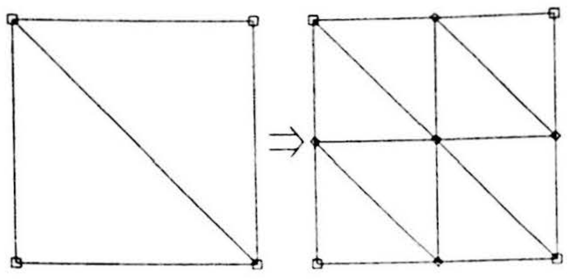


Figure 1 Triangle Edge Subdivision

Triangular edge subdivision is illustrated in Figure 1. Each triangle is divided into four new ones. The edges are each divided equally into two. The midpoint then has a Gaussian random variable added to it. The standard deviation of this is given by:

$$S=k2^{-iH}$$

Where i is the iteration level, k is a scale factor and H is the fractal dimension.

No information is passed between adjacent triangles, so this may be termed to be a 'context independent' method [18]. This is important since it leads to what is known as the creasing problem. An important measure of a randomising interpolant is what it looks like if the randomisation is turned off, i. e. k is set to zero. For the sake of later comparison a simple test case is considered. The fractals are defined to be periodic and to have their control points lying on a square matrix. The vertices of the square and the edge midpoints are all set to a constant level. The midpoint of the square is raised above the constant level.

Image 1a illustrates the effect of subdivision for k equal to zero. A pyramid is the result. This is because below the first level of subdivision each triangle is merely divided into other triangles which are coplanar with it. Thus rather than rolling hills we have 'rolling pyramids' which are unsatisfactory for terrain. Also note that the original control points had square symmetry whereas the surface is skewed. The skewness can be overcome by placing the control points on an equilateral triangular mesh rather than on a rectangular one, but the discontinuities in slope will remain.

Image 1b indicates a plan view of a randomised version of the pyramid with H equal to 1.0. The shading model used is a simple scaling of the X-component of the surface normal. The creases parallel to the X and Y axes and along the diagonal are all too apparent. However, rotated by 60 degrees about the X-axis with Cosine Law shading the mountains look quite respectable and very detailed. See Image 1c. Indeed the creasing artifacts do much to make the mountains dramatic. However it would be more desirable to have random detail generated which could be considered to look natural from all directions.

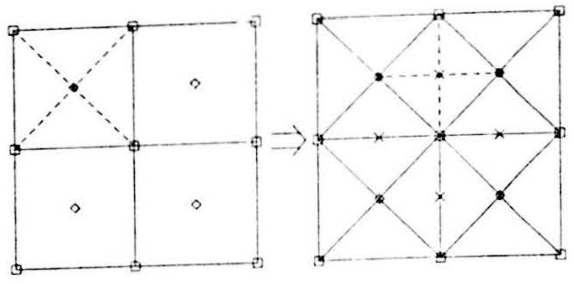


Figure 2 Diamond - Square Subdivision

The second method in [9] was the diamond-square lattice illustrated in Figure 2. Rather than subdividing only edges a square is used to generate its centre point which then leaves holes which are surrounded by data in a diamond arrangement. The diamond is used to generate its centre point and this level of subdivision is complete. This method of subdivision does take values from neighbouring regions and so it is 'context dependent'. However, because of the two tier iteration scheme artifacts do occur. Again with the randomisation turned off Image 2a indicates the rather peculiar surface which results. Note that the original control points have the surface passing through them. The surface is very pointed towards the peak and there are bumps and dents in the surface depending on the local curvature. These are, of course, innate in the interpolant and nothing to do with fractal randomisation which is not present. Image 2b again indicates the X-component of the surface normal. Tell-tale vertical streaks indicate a persistent creasing problem and Image 2c shows pointed peaks not only at the control points but also at other intermediate points as well.

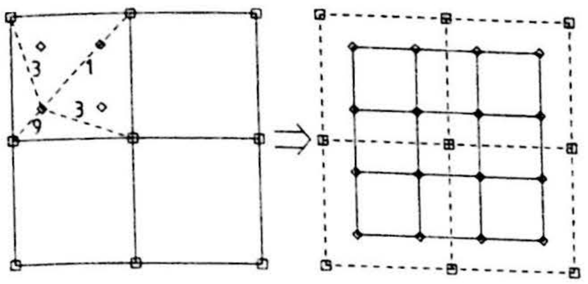


Figure 3 Square - Square Subdivision

The new method here proposed is one adapted from the field of CADCAM. In order to gain first order continuity in the interpolant we sacrifice the requirement of having the surface passing through the control points. Figure 3 illustrates the square-square subdivision rule of [4] and [7]. The new points are generated on a square which is half the size of an existing square. The new values are taken in the proportion 9:3:3:1, the nearer points having the greater weighting. This leads to an interpolant which in the limit is a biquadratic surface. This is a surface which is smooth and continuous in surface normal. Note that with this method the control points deflect the surface but do not normally lie on it.

Image 3a indicates the form of the interpolant. First order continuity is achieved but the surface is rather conservative in the way it is deflected by the control points. Image 3b shows no creasing artifacts and Image 3c is well behaved. However the surface looks rather bland compared with Image 1c, but the value of H can always be decreased to create more roughness, and the control points may be exaggerated in Z to make a more pronounced peak. With this square-square fractal method we may generate terrain which is anything from rolling hills to rough mountains, since the roughness is due to the fractal statistics and is not produced by the underlying interpolant.

Whilst so far we have only considered the use of fractals for terrain, the generation of cloud filled skies is also an important problem. As demonstrated later the new fractal method leads to the generation of acceptable clouds, because of the lack of unnatural artifacts. It is worth noting, however, that for texturing purposes, fractal subdivision methods may be replaced by Perlin's stochastic modelling approach [17].

3. A Parallel Algorithm for Rendering Height Fields

For the purposes of this paper a height field is a surface which in object space coordinates is defined by a single value of Z for every X and Y. We have a viewpoint $\underline{P}$ and a view direction $\underline{V}$. We wish to be able to compute an image of the height field for any view direction and position. The usual approach to this is to consider a plane image whose surface is perpendicular to the view direction vector. A 4x4 transformation matrix is sufficient to project from the object space into the image space. The surface elements are transformed into perspective and then rendered into the image. The use of occlusion-compatible ordering allows the elements to be rendered from back to front with pixel overwrite, or from front to back for antialiasing [5]. A general polygon-based system explicitly computes this order from the polygon data, or derives it from a precomputed data structure [11]. With height fields, however, we may divide the surface into four rectangular regions within which the ordering can be generated trivially. We split the object X-Y plane into four regions using lines parallel to the X-axis and Y-axis which intersect vertically beneath the viewpoint. Each region may then be rendered using polygons generated in the correct row by row or column by column order [1]. We then need a fast method of scan-conversion, but this may be difficult to distribute evenly between parallel processors. In particular, a height field in perspective may not project to singly valued screen y coordinates. The importance of this is explained later. Another problem with this method is that if several different view directions are required, then the visibility calculations must be recomputed. Similarly it is difficult to process scenes for which the view angle is greater than 180 degrees.

An established technique for the projection of planar textures into perspective is to use a row-column access method. In this approach pixels are read, transformed and stored for each row in turn. The columns are then each accessed and transformed. The net result is a planar texture which has been transformed into perspective with antialiasing [3]. This scheme may be thought of as a parallel algorithm since each row could be processed by a different processor at the same time. Each column could then be processed similarly. The advantage of this method is that there is no executive processor allocating fragments of the entire image to each subprocessor. Also each processor does not need to store the entire image, but only a single row or column. The difficulty with this method is the dual access of the data first for the rows and then for the columns. If the

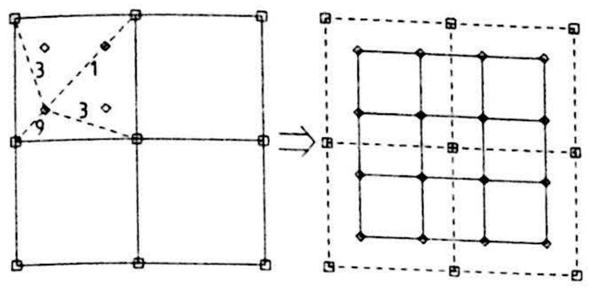


Figure 3 Square - Square Subdivision

The new method here proposed is one adapted from the field of CADCAM. In order to gain first order continuity in the interpolant we sacrifice the requirement of having the surface passing through the control points. Figure 3 illustrates the square-square subdivision rule of [4] and [7]. The new points are generated on a square which is half the size of an existing square. The new values are taken in the proportion 9:3:3:1, the nearer points having the greater weighting. This leads to an interpolant which in the limit is a biquadratic surface. This is a surface which is smooth and continuous in surface normal. Note that with this method the control points deflect the surface but do not normally lie on it.

Image 3a indicates the form of the interpolant. First order continuity is achieved but the surface is rather conservative in the way it is deflected by the control points. Image 3b shows no creasing artifacts and Image 3c is well behaved. However the surface looks rather bland compared with Image 1c, but the value of H can always be decreased to create more roughness, and the control points may be exaggerated in Z to make a more pronounced peak. With this square-square fractal method we may generate terrain which is anything from rolling hills to rough mountains, since the roughness is due to the fractal statistics and is not produced by the underlying interpolant.

Whilst so far we have only considered the use of fractals for terrain, the generation of cloud filled skies is also an important problem. As demonstrated later the new fractal method leads to the generation of acceptable clouds, because of the lack of unnatural artifacts. It is worth noting, however, that for texturing purposes, fractal subdivision methods may be replaced by Perlin's stochastic modelling approach [17].

3. A Parallel Algorithm for Rendering Height Fields

For the purposes of this paper a height field is a surface which in object space coordinates is defined by a single value of Z for every X and Y. We have a viewpoint $\underline{P}$ and a view direction $\underline{V}$. We wish to be able to compute an image of the height field for any view direction and position. The usual approach to this is to consider a plane image whose surface is perpendicular to the view direction vector. A 4x4 transformation matrix is sufficient to project from the object space into the image space. The surface elements are transformed into perspective and then rendered into the image. The use of occlusion-compatible ordering allows the elements to be rendered from back to front with pixel overwrite, or from front to back for antialiasing [5]. A general polygon-based system explicitly computes this order from the polygon data, or derives it from a precomputed data structure [11]. With height fields, however, we may divide the surface into four rectangular regions within which the ordering can be generated trivially. We split the object X-Y plane into four regions using lines parallel to the X-axis and Y-axis which intersect vertically beneath the viewpoint. Each region may then be rendered using polygons generated in the correct row by row or column by column order [1]. We then need a fast method of scan-conversion, but this may be difficult to distribute evenly between parallel processors. In particular, a height field in perspective may not project to singly valued screen y coordinates. The importance of this is explained later. Another problem with this method is that if several different view directions are required, then the visibility calculations must be recomputed. Similarly it is difficult to process scenes for which the view angle is greater than 180 degrees.

An established technique for the projection of planar textures into perspective is to use a row-column access method. In this approach pixels are read, transformed and stored for each row in turn. The columns are then each accessed and transformed. The net result is a planar texture which has been transformed into perspective with antialiasing [3]. This scheme may be thought of as a parallel algorithm since each row could be processed by a different processor at the same time. Each column could then be processed similarly. The advantage of this method is that there is no executive processor allocating fragments of the entire image to each subprocessor. Also each processor does not need to store the entire image, but only a single row or column. The difficulty with this method is the dual access of the data first for the rows and then for the columns. If the processors are connected by a common bus then the image may be flipped about a diagonal in one frame time.

It would be attractive to apply a row-column algorithm to the rendering of height fields. The rows would be stretched laterally in accordance with the perspective projection and then the columns would be transformed to the correct view. Unfortunately this approach is only exact for perspective projections if the view vector is parallel to the X-Y plane. For other view directions the projected height field is not singly valued in screen y coordinate. Indeed the process collapses if the view vector is straight down along the object space Z-axis. However this inexact approach has been used to good effect in [10].

The new approach presented in this paper is to abandon the planar projection and to use an intermediate viewing sphere. (This is an extension of the method described in [8] which was restricted to having view directions parallel to the X-Y plane). We consider the viewpoint to be surrounded by a sphere onto which have been projected the surfaces from all directions. This sphere may then be transformed into screen coordinates for viewing on a flat screen or directly projected onto a spherical screen for wide angle work.

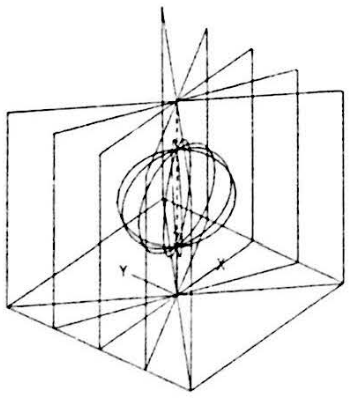


Figure 4 Viewing Sphere with Axial Planes

The advantage of this method is that the visible surface calculation for the viewing sphere is independent of the viewing direction and the processing may use planes which pass vertically through the terrain map. See Figure 4. If we consider the family of planes which pass through a line dropped vertically from the viewpoint, then the planes will have intersections with the terrain which are singly valued in Z.

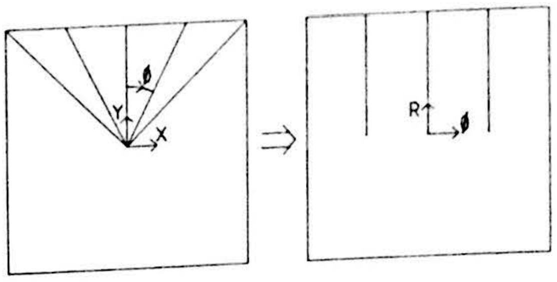


Figure 5 Cartesian to Polar Conversion

A way to achieve this with a row-column algorithm is to split the object plane into wedge quadrants. These are + or - 45 degrees from either the X or Y axes. We stretch each row until the vertical planes of constant Phi become columns. See Figure 5.

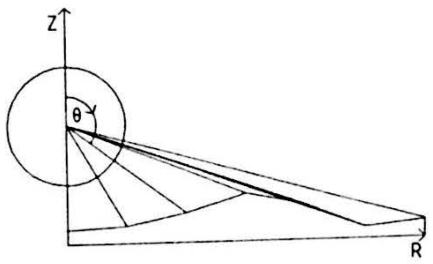


Figure 6 Projection of Slice onto Viewing Sphere

To project onto the viewing sphere we then process each column using inverse trigonometric functions to compute values of Theta. See Figure 6. From a Phi-Theta representation of the viewing sphere we proceed to project onto the viewing plane. See Figure 7.

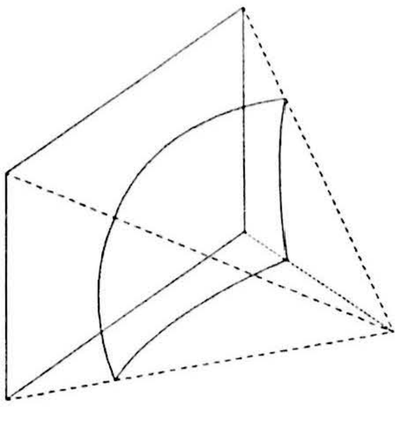


Figure 7 Projection of Viewing Sphere onto Image Plane

Image 4 illustrates this process. Top left is the original terrain map with areas within the wedge quadrants of interest highlighted in grey. Top right is the same terrain after it has been expanded row by row in X by a factor of 1/Y. Bottom left is the same region after a subsequent column stretch by a factor of 1/Cos(X), where X has been scaled to lie between -Pi/4 and Pi/4. Note that the bottom half of the data was then flipped vertically. The top and bottom halves now correspond to two opposite wedge quadrants. Bottom right is a polar representation of half of the viewing sphere. Phi is along X, Theta is along Y. The two wedge quadrants have been projected onto the viewing sphere. The lack of visual detail for near parts of the terrain is readily apparent.

Image 5 demonstrates the additional use of a two-dimensional version of Perlin's fractal texturing technique. In this a bandwidth-limited function of X and Y is successively scaled and superimposed in accordance with the viewing transformation. This is used to compute a normal perturbation for the shading function. With this near elements remain detailed despite their expansion by the spherical projection. Top right is an orthogonal spherical projection taken along the Y-axis. Top left is the same projection taken looking straight down along the Z-axis. Image 6 shows a perspective projection of the viewing sphere taken along the X-axis.

If the viewing sphere is transformed such that the polar axis lies along the current rotation axis of the observer, then rotational motion blur may be computed. A block or ramp filter is applied along the Phi direction of the polar representation of the viewing sphere. Image 7 is the same scene as that shown in Image 6, but with motion blur due to a barrel roll about the view vector. Image 8 shows a wide angle projection of the viewing sphere after it has been tilted about the X-axis by 45 degrees. Image 9 is a perspective projection of the rotated viewing sphere.

The details of the row-column algorithms for arbitrary spherical-to-spherical and spherical-to-planar transformations will be the subject of a later work [15]. However it should now be apparent that by using a spherical projection as an intermediate representation it is possible to construct a row-column algorithm for the perspective rendering of height fields.

4. The Fan-tracing of Height Fields

Recently some work has been done on ray-tracing fractal surfaces defined by the triangle edge subdivision technique [2] and [14]. While these methods work they are slow and restricted to a certain class of surfaces. In general the key uses in terrain modelling for ray-tracing like effects would be the calculation of shadows and reflections in water. Usually where shadows are required the sun may be considered to be a point light source. This is equivalent to computing a second hidden surface calculation. A more difficult problem is the reflection of light in water. For distant regions there may be many water waves per pixel. Thus many rays per pixel will be required for raytracing rippling water. Also each ray must be separately traced against the whole terrain.

The approach adopted here is to assume that rays may be perturbed vertically by the water but not laterally. This allows us to model the actual appearance of reflections of distant hills in lakes quite realistically whilst reducing a 3-D ray trace to a 2-D one. Since these vertical planes are equivalent to those in the previous section this procedure may be implemented in parallel as a natural part of the hidden surface algorithm. Since reflected light is scattered vertically but not horizontally, the light which will be scattered towards an observer lies within a range of vertical angles from the point on the water surface. This may be thought of as a 'fan' of light rays incident on the water. The reflected light intensity is merely the integral of the incident light lying within the range of the fan.

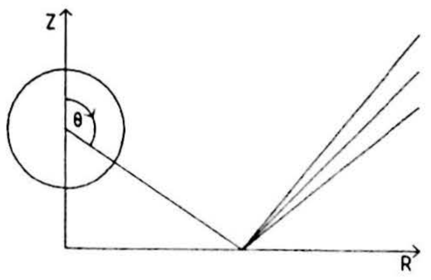


Figure 8 The Beam Geometry of Water Reflection

To ease the computational load the water is treated as flat with a normal perturbation. The waves are modelled as superimposed sine waves which aids clamping, i. e. the bandwidth-limiting of the normal perturbation texture [16]. However since we are interested in reflections of a fan rather than the projection of a texture the waves have two effects. The first is to perturb the centre of the fan based on the slope of the wave surface. The second is to spread the fan based on the curvature of the wave. See Figure 8. The fan centre perturbation is clamped to zero for distant waves whilst the fan spread is set to maximum. An alternative to superimposed sine waves would have been the wave model in [17]. Perlin's model uses several point sources of disturbance and so avoids the periodicity inherent in the sine wave approach. It gives very realistic pictures and the incorporation of that method into this system is an area for future work.

If there were none of the height field protruding above the water level then the intensity for a fan could be computed from an environment map of the sky. For each vertical slice this environment map would be one-dimensional, and a precomputed integral table could be used to speed calculations [6]. However, since the height field can obstruct all or part of the sky contribution to the fan, a more complicated approach is required.

For each water pixel a fan is generated based on the projection distance, the perturbation and the spread of the wave. The plane slice of the height field is projected and clipped to that fan and the intensities are averaged over the fan. Thus there is one fan per water pixel rather than many rays. For Image 10, the view direction was taken parallel to the negative Y-axis. Also the height field was painted onto piecewise flat vertical elements for rapidity of projection. This latter simpification did lead to unnatural vertical streaks for some images and is not recommended. However the results were quite encouraging. The clouds were generated using the new fractal method as were the mountains. The snow line was prevented from being too regular by the use of normal as well as height data to determine the snow threshold.

By dividing a 3-D problem into a 2-D problem a great speed saving was achieved thus allowing the simulation of quite realistic effects on a small machine. Image 10 took 6 hours on a Prime 250 for 512x512 resolution, including the time for the generation of the terrain data.

5. Conclusions and Future Developments.

This paper has presented a new method for the generation of fractals by recursive subdivision, which does not create the artifacts of previously published methods. It has also presented a parallel algorithm for the perspective rendering of height fields which uses an intermediate spherical projection. This algorithm was extended to include an approximate but convincing method for the simulation of reflections of terrain and sky in water.

The common bus architecture mentioned in this paper may be replaced by a more complex one, to speed up the diagonal flip process.

Also, the height field rendering algorithm as given does not allow the inclusion of collections of trees or bushes. By allowing more data to be stored in the rows and columns it may be possible to include features such as the textured ellipsoids of Gardner [12] and [13].

Acknowlegments

Thanks to Jon Hunwick who designed the yacht in Image 10 on the DUCT surface modeller. Thanks to my Ph. D. supervisor Dr. D. P. Sturge for encouragement and support and to Deltacam Systems Ltd and Sigmex Ltd for the loan of equipment.

References

[1] Anderson D. P., Hidden Line Elimination in Projected Grid Surfaces, A.C.M. Trans. Graphics, Vol. 1, No. 4, Oct 1982, pp 274-288.

[2] Bouville C., Bounding Ellipsoids for Ray-Fractal Intersection, SIGGRAPH '85, Computer Graphics, Vol. 19, No. 3 (July 1985).

[3] Catmull E. and A. R. Smith, 3-D transformations of images in scanline order, Computer Graphics, Vol. 14, No. 3, pp 279-284, 1980.

[4] Catmull E. and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, CAD Vol. 10, pp 350-355, 1978.

[5] Coquillart S. and M. Gangnet, Shaded Display of Digital Maps, IEEE Computer Graphics and Applications, Vol. 4, No. 7. July 1984.

[6] Crow F. C., Summed-area tables for texture mapping. Computer Graphics, Vol. 1, No. 3 (July 1984), pp 207-212.

[7] Doo D. and M. Sabin, Behaviour of recursive division surfaces near extraordinary points, CAD Vol. 10, pp 356-362, 1978.

[8] Fishman B. and B. Schecter. Computer Display of Height Fields. Computers and Graphics Vol. 5 (1980) pp53-60.

[9] Fournier A., D. Fussell, L. Carpenter, Computer Rendering of Stochastic Models. Comm. of the A.C.M., 25, 6, (June 1982), pp 371-384.

[10] Fournier A. and T. Milligan, Frame Buffer Algorithms for Stochastic Models. IEEE Computer Graphics and Applications. October 1985. Vol. 5, No. 10.

[11] Fuchs H., G. D. Abram and E. D. Grant. Near Real-Time Shaded Display of Rigid Objects, Computer Graphics, Vol. 17, No. 3 (July 1983).

[12] Gardner G. Y., Simulation of Natural Scenes Using Textured Quadric Surfaces. Computer Graphics Vol. 18, No. 3 (July 1984) pp 11-20.

[13] Gardner G. Y., Visual Simulation of Clouds, Computer Graphics, Vol. 19, No. 3 (July 1985), pp 297-303.

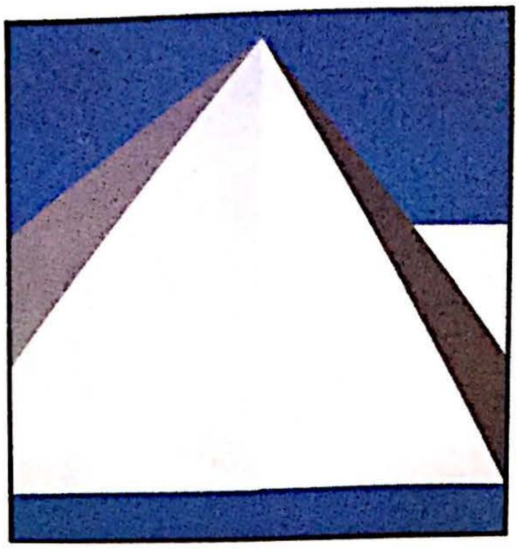
[14] Kajiya J. T., New Techniques for Ray Tracing Procedurally Defined Objects, Computer Graphics, Vol. 17, No. 3, (July 1983).

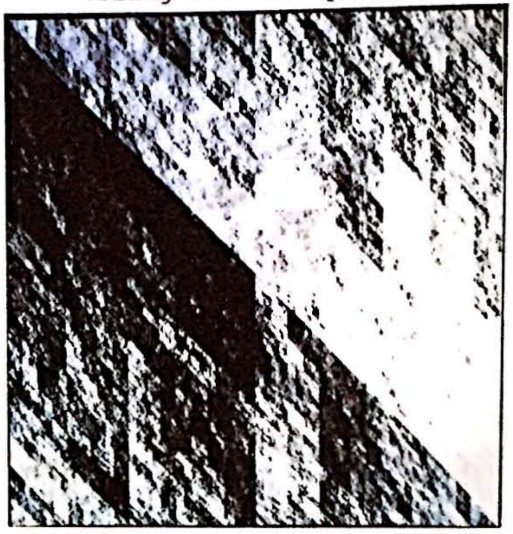
[15] Miller, G. S. P., Author's unpublished Ph. D. dissertation - work in progress.

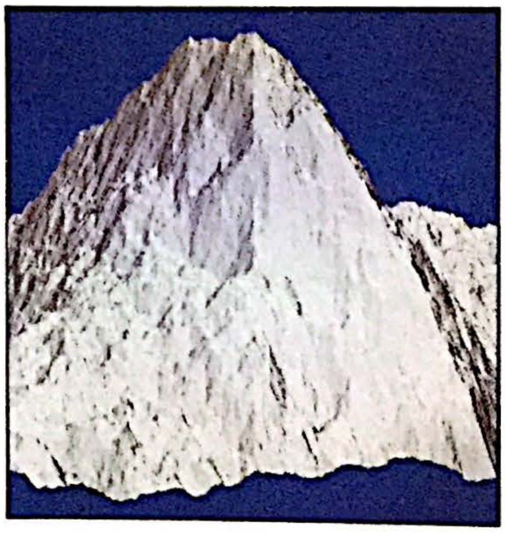
[16] Norton A., A. P. Rockwood and P. T. Skolmoski, A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space. Computer Graphics Vol. 16, No. 3 (July 1982).

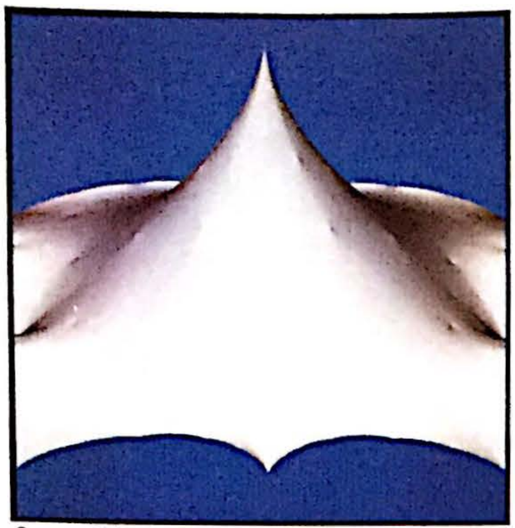
[17] Perlin K., An Image Synthesizer, SIGGRAPH '85, Computer Graphics, Vol. 19, No. 3 (July 1985).

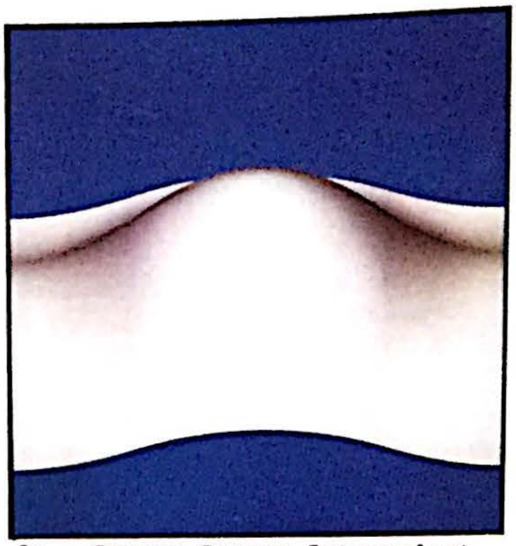
[18] Smith A. R., Plants, Fractals and Formal Languages. Computer Graphics, Vol. 18, No. 3 (July 1984).

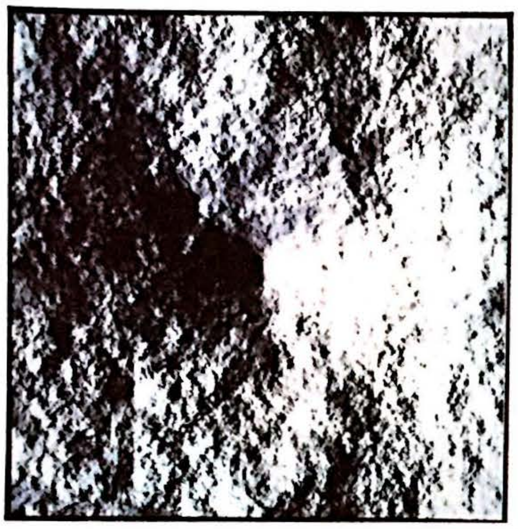

Triangular Interpolant.

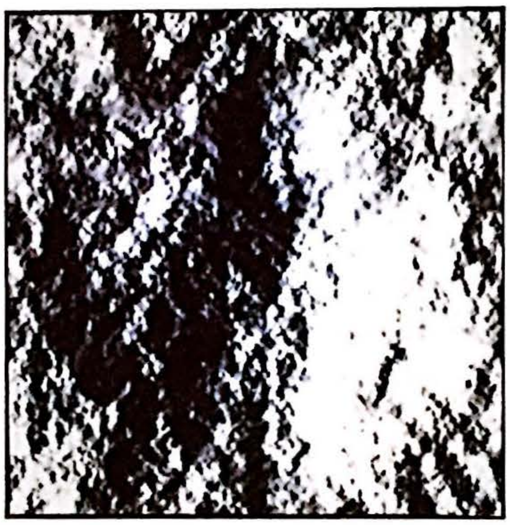

1b. Triangular Fractal Normal Map.

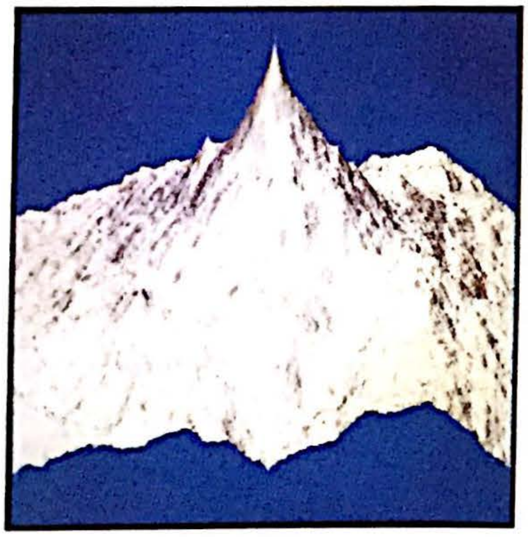

1c. Triangular Fractal Mountain.

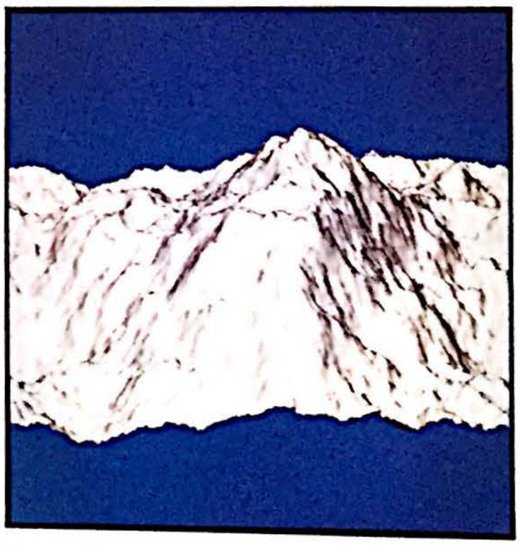

2a. Diamond-Square Interpolant.

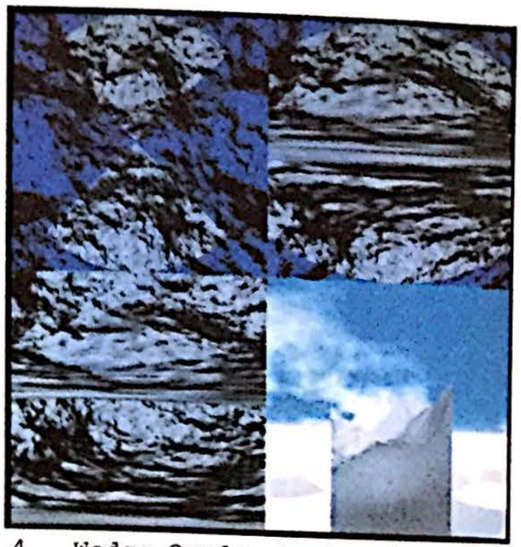

3a. Square-Square Interpolant.


2b. Diamond-Square Fractal Normal Map.

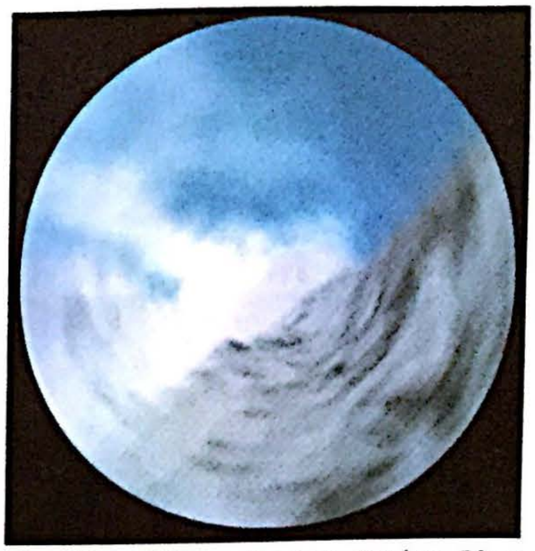

3b. Square-Square Fractal Normal Map.


2c. Diamond-Square Fractal Mountain.

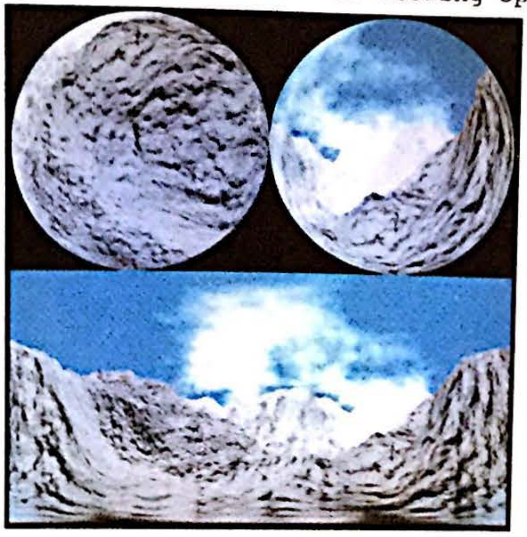

3c. Square-Square Fractal Mountain.



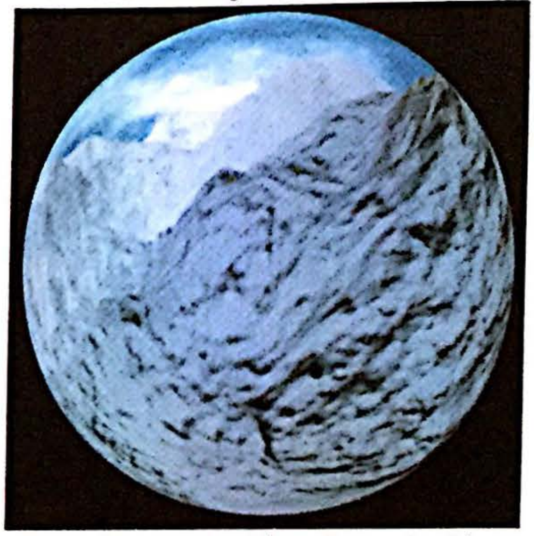
4. Wedge Quadrants to Viewing Sphere.



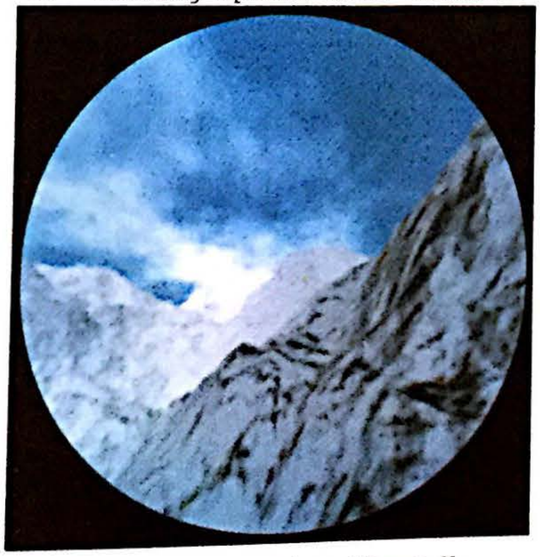
7. View Along X with Motion Blur.



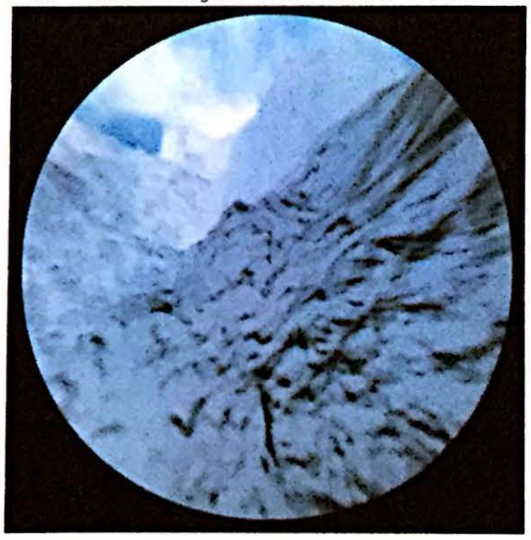
5. Viewing Sphere With Texture.



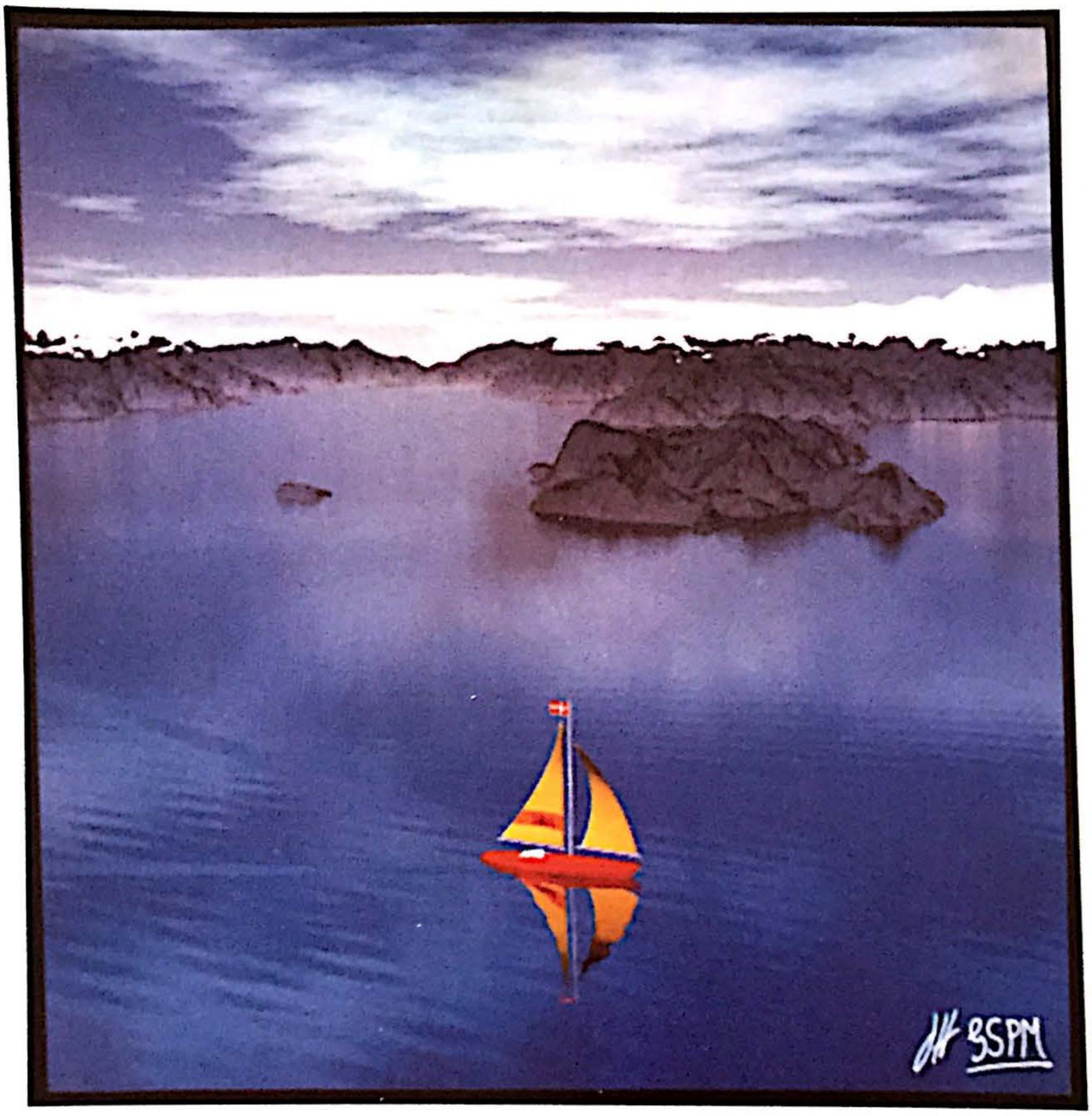
8. Wide Angle View Down Incline.



6. Perspective View Along X.



9. Perspective View Down Incline.



10. Sailing By.