A New Fractal-Based Approach for 3D Visualization of Mountains in VRML Standard

Mohsen Sharifi, Fatemeh Hashemi Golpaygani., Mehdi Esmaeli and Javad Sadeghi

Visualization Research Group
Computer Engineering Department
Iran University of Science and Technology
{msharifi, f_hashemi, m_esmaeili, javad} @iust.ac.ir

Abstract

Several factors currently limit the size of Virtual Reality Modeling Language (VRML) models that can be effectively visualized over the Web. Main factors include network bandwidth limitations and inefficient encoding schemes for geometry and its associated properties. The delays caused by these factors reduce the attractiveness of VRML usage for a large range of virtual reality models, CAD data, and scientific visualizations. To solve this problem, we have tried to decrease the size of data by deploying *fractal geometry* in VRML standard. A novel approach is proposed for generating a "fractal mountain" using a random Midpoint-Displacement method in VRML standard. Our VRML 2.0 implementation, which is based on two newly defined nodes, *TriangleGrid* and *FractMountain*, and uses PROTO mechanism and Java in the Script nodes for the logic, is presented too. It is shown that our approach is more flexible and memory efficient than other approaches for computing mountain structures. Besides, mountains visualized by this approach look much more *natural* than those generated by other approaches.

Keywords: Fractal, Midpoint Displacement, 3D Visualization, Virtual Reality, VRML

1 Introduction

A geometric approach to represent territory and landscape is to determine 3D objects, which in turn consists of finding a model that represents a set of data points. A wide variety of representation methods have been studied for modeling these surfaces [Bolle et al. 1991] (For example, "Lorentz hat"). Unfortunately, these models do not produce rough surfaces. Models that are able to produce rough surfaces are mostly based on random processes, and for this reason these models are not suitable for approximation. To solve the problem of rough surface approximation efficiently fractal geometry has been used.

Before the development of fractal geometry, typically Nature was regarded as "noisy" Euclidean geometry. A mountain is primarily a roughened cone, for example. Indeed, this view was codified by Paul Cezanne's statement about painting: "*Everything in Nature can be viewed in terms of cones, cylinders, and spheres*". In contrast to this, Benoit Mandelbrot asserts, "*Clouds are not spheres, mountains are not cones, coastlines are not circles, and bark is not smooth, nor does lightning travel in a straight line*" [SALA et al. 2002].

The word "fractal" was coined less than thirty years ago by one of the history's most creative mathematicians, Benoit Mandelbrot [Mandelbrot 1983], father of fractal geometry, but only with the mathematical power of computers it has become possible to derive beautiful and colorful images out of complex formulas.

A fractal object is self-similar in that subsections of the object are similar in some sense to the whole object. No matter how small a subdivision is taken, the subsection contains no less detail than the whole.

Fractals have found their way into computer graphics, image and video compression, and even film and music. For example Hollywood used fractals in the movie "Star Trek II: The Wrath of Khan". They have become a way of modeling real and virtual aspects of the world [Howe 2000].

The advantage of displaying and rendering fractal graphics is that they can be summed up in a very few rules. For example a terrain displaying a complex object like a mountain range would only require a few rules to be described as a fractal. However to store all the points of the terrain would take up an enormous amount of space. This storage saving advantage has the disadvantage of requiring heavy computing power to generate such complex objects [Howe 2000].

In order to get a good visual effect and effective query, the display of geographical data should be in a 3-D mode and capable of being interacted. This goal can be reached by integrating advances in computer graphics and Internet technology. Advanced graphical libraries, such as VRML (Virtual Reality Modeling Language) [Carey and Bell 1997], make it possible to effectively model and thereafter render the third dimension.

Since its first release in early 1995, VRML has gained broad support. It is widely used today for scientific visualization, architectural models, simulations, and even 3D cartoons. The VRML 97 became an ISO (International Organization for

Standardization) standard in December 1997 and now is in the Moving Picture Expert Group's new MPEG-4 standard [Nadeau 1999]. VRML has 54 different types of nodes available, plus the ability to define new node types that extend the language. Parallel Graphics has produced a popular VRML browser, called *Cortona*, which is provided as a plug-in for Netscape and Internet Explorer.

Web3D Consortium has recently introduced an open and extensible standard (X3D) as the next generation of VRML, wherein the geometry and behavioral capabilities of the VRML 97 standard are mainly complemented with the eXtensible Markup Language (XML) [Web3D 2003].

The visualization research group at Computer Engineering Department of Iran University of Science and Technology is aiming to work out the design and implementation of standards in VRML (and X3D) to visualize varieties of natural phenomena that are modeled by Fractals. In this paper a part of this activity related to the efficient visualization of Fractal *Mountains* in VRML through the definition of two *new VRML nodes* is reported.

VRML and Java have been chosen for programming (Section 2). New nodes have been defined using the extensibility feature of VRML (Section 3). Different methods of creating mountains were investigated (Section 4) and an approach leading to the definition of ***TriangleGrid*** and ***FractMountain*** nodes and their logic were proposed (Section 5) and evaluated (Section 6). Further works on the subject were clarified finally (Section 7).

2 VRML and Java Combined

VRML and Java provide a standardized, portable and platform independent way to render dynamic, interactive 3D scenes across the Internet. Integration of these two powerful and portable software languages gives us interactive 3D graphics and complete programming capabilities plus network access. Interfaces between VRML and Java are made through Script nodes.

VRML provides the 3D scene graph, Script nodes encapsulate Java functionality, and ROUTEs provide the wiring that connects computation to rendering. Script nodes appear in the VRML file, encapsulating the Java code and providing naming conventions for interconnecting Java variables with field values in the scene. Java classes for use with the Script node must import the vrml package.

Using Script nodes in VRML 2.0, we can use Java classes to affect the VRML world. With Script nodes, we are essentially creating a new node, with associated fields and events.

A great deal of implementation work is now in progress. The most optimistic view is that VRML and Java are well matched, well specified, openly available and portable to most platforms on the Internet [Brutzman 1998].

3 Extensions to VRML

In VRML 2.0, we can encapsulate a group of nodes together as a new node type, a prototype, and then make that node type available to anyone who wants to use it. We can then create instances of the new type, each with different field values. EXTERNPROTO definition specifies the remote URL where the original PROTO is defined. Using the PROTO mechanism of VRML and a Script node executing Java code, one can implement a new VRML node [Nadeau 1999]; what we have exactly done for the definition of our proposed nodes.

GeoVRML [GeoVRML 2003], which is an extension to the ISO standard VRML for providing support for geographic applications, is a notable example that has used the EXTERNPROTO property and the Script mechanism to define its nodes.

GeoVRML 1.0 offers the unique capability of an open file format that lets geoscientists integrate their graphic data directly into a 3D computer graphics scene and allows remote users to view the result interactively over the web using freely available standards-based software. It was publicly announced as an official recommended practice of the Web3D Consortium in June of 2000 and includes ten new extensions, or nodes, that sit on top of VRML97. These nodes are defined using VRML's EXTERNPROTO extensibility features.

4 Fractal Mountains

There are several methods to create mountains. One of them is Lindenmayer systems (L-systems). Most work on L-systems is entirely focused on modeling of plants but this method can be used also for modeling of mountains. This method is very flexible but at the same time has limitation in size, because we have to be careful so that the shape does not get too high or low since this will create some extreme peaks [Johnson 1999].

Another method to build a mountain with fractal methods is the application of midpoint displacement algorithm introduced by Benoit Mandelbrot [Mandelbrot 1983]. In one-dimensional midpoint displacement, we start with a line segment and then recursively, for each line segment, find and displace the midpoint by a random amount along the Y axis (vertically). This work continues until reaching the desired level of detail, reducing the random amount on each iteration.

A realistic mountain is three dimensional (3D), which requires midpoint displacement in two dimensions. There are three efficient methods that can be used to create 3D mountains using midpoint displacement, with *triangle*, *square* and *hexagonal grids.* Triangles have simple calculations [Fournier et al. 1982]; each side of the triangle is split half way and together these three new points split the original triangle up into four smaller triangles.

But Miller believes [Miller 1986] that triangle method is '*content independent'* meaning that for adjacent triangles, no information is passed thus leading to the creasing problem. He further believes that squares method avoids this problem to some extend by offsetting the grid during decomposition. The height of each new vertex depends on all the neighboring vertices. The overhead is quite large when the above recursive subdivision is used.

Mandelbrot introduced hexagon algorithm to generate terrain *without creases* or seams [Peitgen and Saupe 1988]. Hexagons finally solved the creasing problem by using a more complex system. During decomposition, each hexagon is broken up into three smaller hexagons. The height of additional vertices for the smaller hexagons is averaged from neighboring vertices. Unfortunately, Hexagon algorithm is almost inefficient because it needs more space than the other two ones.

Another method to create mountains is Diamond-Square algorithm. This algorithm was originally described by Fournier, Fussell, and Carpenter [Fournier et al.1982]. It uses the edges to generate center points to be used as edges of other squares/diamonds rather than just subdividing the edges.

This method of subdivision takes values from neighboring regions and so it is *'context dependent'*. The size of generated facets of this algorithm can get very large in a small number of iterations. Miller argues that this Diamond-Square method is also susceptible to the creasing problem [Miller 1986].

According to Miller, another problem with this method is that the generated scene is not realistic as it artificially inputs the height number for the center point (the peak) to force the generation of a mountain, but Paul Mart does not agree with Miller on this point by saying that if all the other height points are randomly generated then why not do the same with the center peak? [Mart 2002]

It can therefore be concluded that, out of methods generating Fractal Mountain, the midpoint displacement algorithm on triangle grid provides a more flexible and memory efficient way of computing mountain structures.

5 Our Proposition: *TriangleGrid* and *FractMountain* Nodes

We have created 3D mountains using the midpoint displacement method on a Triangle grid in VRML standard. The grid that is being used to create the mountain is represented as a collection of triangle objects.

The ElevationGrid node[1] has mostly been used for the creation of mountains in VRML standard, requiring the collection of huge amount of data about the height of every single point in the whole grid.

We have decided to generate the "Fractal Mountain" randomly using fractal midpoint displacement method, instead of using a height field, in order to reduce the size of data. We want to introduce two new nodes into VRML standard: TriangleGrid and FractMountain.

Given the advantages of creation of Fractal Mountain on triangle grid, and considering the absence of such a node in VRML standard, we had to define the new *TriangleGrid* node. Otherwise, we had to simulate midpoint algorithm on some other grid unsatisfactorily, like others had done before. For example, Demidov [Demidov 2003] has simulated the midpoint displacement method on ElevationGrid, but his resulting mountain does not look very natural.

For creation of triangle grid in VRML, we used the *IndexedFaceSet* node in VRML. TriangleGrid node has two fields: the first field is "n" and the second one is "height". The first field, "n", shows the size of grid, and the "height" represents an array that contains all the heights of every single point in the grid. The number of points in the grid is (n (n+1))/2.

[1] The ElevationGrid node describes a uniform rectangular grid of varying height. The grid lays in the y=*0* plane, and a scalar array of heights determines the height values of the *y*-axis.

Each point is a 3D object that represents a set of coordinates (x, y, z). The height of the grid is initially zero (y = 0), but the x and z have to be calculated. The index of each point is as shown in Figure 1.

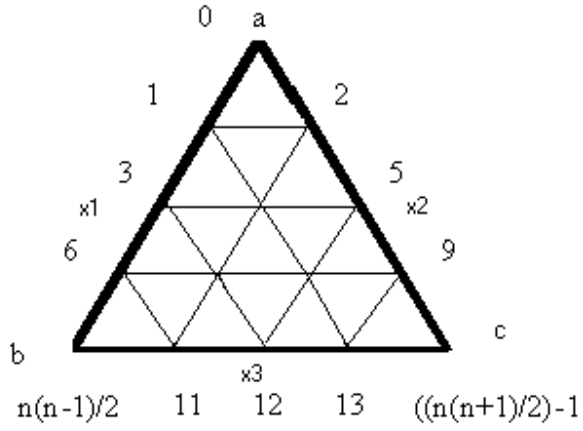


Figure 1. Triangle Grid

For example, for index 0, the set of data point is (n-1, 0, 0). After all the coordinates and indexes have been created for each point, we can introduce the triangles of the node, using different groups of three indices.

Our second proposed node is *FractMountain*, which takes (TYPE, N, ROUGHNESS) triple as input and outputs the relevant mountain. The first field, TYPE, declares the kind of algorithm that has to be used, as well as the type of grid (i.e. TriangleGrid or ElevationGrid). The second field, N, declares the size of the mountain. The last field, ROUGHNESS, lets the user to have the choice of different surfaces for the mountain (i.e. rough or smooth surfaces). If the TYPE is set to zero, the midpoint displacement algorithm on the triangle grid is used.

Having defined the new node *TriangleGrid*, we needed a new algorithm for applying the midpoint displacement method on this node; the algorithm is nicknamed "*Midpoint Displacement Triangle Grid*" (MDTG).

The MDTG algorithm starts with three basic index points, (0, (n(n+1))/2-1, (n(n-1))/2) and then calculates three new midpoints from them, as is shown in Figure 1. In order to simplify our calculations, an array keeps the level of each point in the grid.

On each level of MDTG algorithm, we find three new middle points of the triangle. If (x1, x2, x3) are three middle points of abc triangle, x1 is set by the algorithm to the number of previous points. Because the number of points in the triangle grid from the start point to level r is ((r+1)(r+2))/2, the number of points before x1 is equal to the amount of "a" plus the amount of the difference between the number of points of level "r" and level "a". Also x2 is equal to x1+(c-b)/2, and x3 is equal to (b+c)/2.

The algorithm finds the height of these middle points (M[x1], M [x2], M[x3]). For example, if "x1" is the middle point of "a" and "b", M[x1] = (M[a] +M[b])/2 + R. The amount of R is calculated using "n", "Roughness" and a random number. To have more

natural mountains, the amount of "R" has to be decreased after every step; otherwise the surface will become very rough.

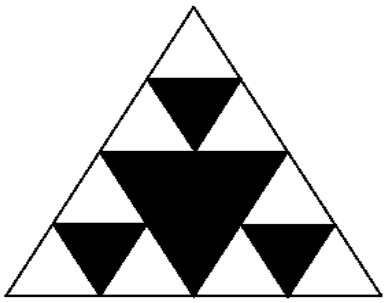


Figure 2. Sierpinski Triangle

Up to now, we have only calculated the points similar to the Sierpinski Triangle points (shown in Figure 2). We have added more levels to MDTG algorithm to calculate the remaining ignored points.

The output of MDTG algorithm is an array M that contains the height of each single point of the triangle grid.

Using MDTG algorithm and PROTO mechanism of VRML and the Script node executing java code, we implemented a new VRML node called *FractMountain*. With this new node, one can create virtual 3D fractal mountains with a very small size of data.

Interestingly, with little syntactic change in the coding, the two proposed nodes are also usable in X3D.

6 Evaluation

Normal methods of generating mountains require a huge amount of data, while fractal methods and VRML require substantially less data. The latter methods too differ in the amount of required data and the quality of their generated mountain. To evaluate these variations, we have compared three methods of generating mountains in VRML standard: (1) mountain created by ElevationGrid, (2) mountain created by midpoint displacement method on ElevationGrid, and our proposed method (3) mountain created by midpoint displacement method on TriangleGrid.

6-1 Mountains Created by ElevationGrid

The mountain created by ElevationGrid is shown in Figure 3. For generating mountain in this method we used huge amount of data to represent the height of each single point. Although there are 169 (13*13) points in the grid, which may seem enough for a natural visualization, the result is not a natural mountain and it has several artificial flat planes.

```
Shape {
 geometry  ElevationGrid {
     xDimension  13
     zDimension  13
     xSpacing   .8
     zSpacing   .8
   height[
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 1.5 0 0 0 0 0 0
0 0 0 0 0 0.5 1 0.5 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 2 1.5 1 2 3 2 1 1.5 2 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0.5 1 0.5 0 0 0 0 0
0 0 0 0 0 0 1.5 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0]}}
```

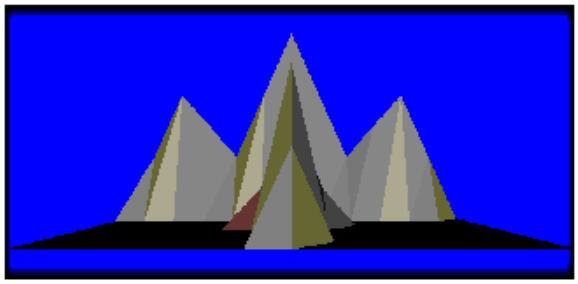
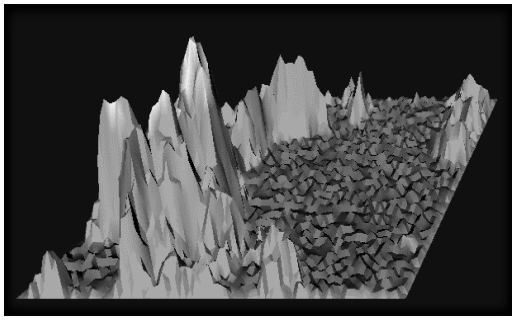


Figure 3. Mountain Created by ElevationGrid

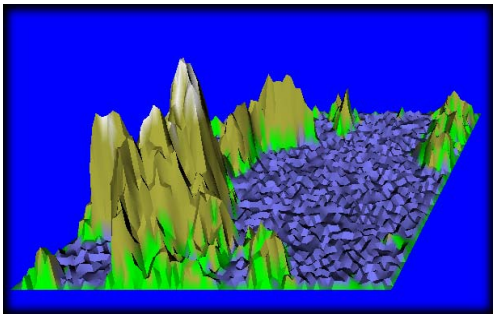
6-2 Mountains Created by Midpoint Displacement Method on ElevationGrid

Demidov has created a terrain by midpoint displacement method on ElevationGrid [Demidov 2003]. In this method, the rectangle view is populated by sea and one or more mountains. He has improved the appearance of generated terrains by painting, for example low height points (having low *y* values) with blue color.

Figure 4 shows a terrain generated by Demidov's method. But it should be noticed that the whole rectangle visualizes *a terrain*-containing mountain, *not just a Fractal Mountain*. This shortcoming is due to the simulation of midpoint displacement method with *triangle* on *rectangle* grid, where the grid is not triangle.



(a)

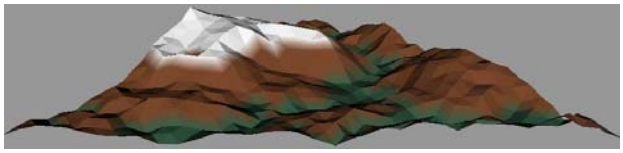


(b)

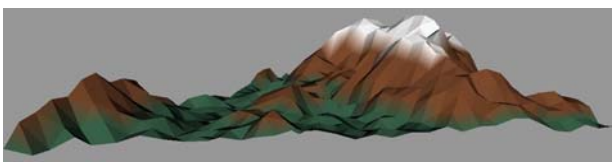
Figure 4. Simulated terrain with n = 64 with midpoint displacement method on ElevationGrid; (a) without painting, (b) terrain vertexes painted in white, brown, green and blue colors according to their *y* values.

6-3 Mountains Created by Midpoint Displacement Method on TriangleGrid

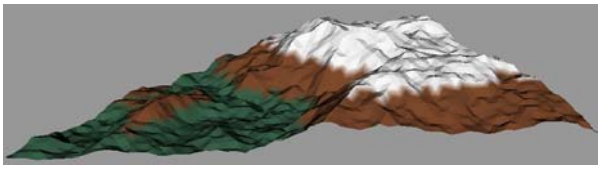
As is shown in Figure 5, the mountain generated by our method, using midpoint displacement on TriangleGrid, gives a more natural visualization of mountain. This is because midpoint displacement method is used on *triangle* grid. An interesting point experienced is that we only needed a small size of data space (1.35 KB) to generate these figures.



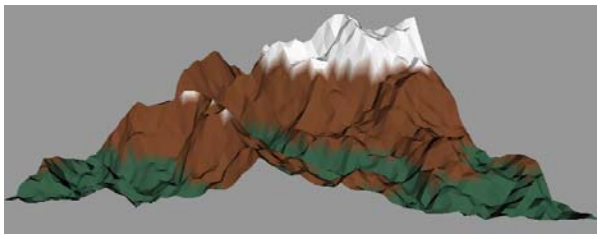
n = 32 roughness parameter = 1



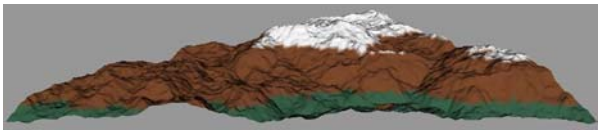
n = 32 roughness parameter = 2



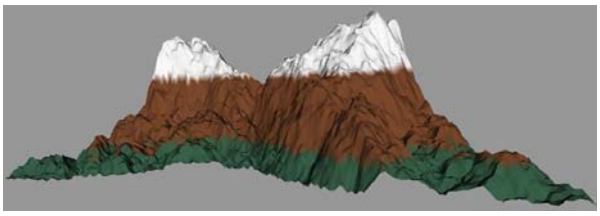
n = 64 roughness parameter = 1



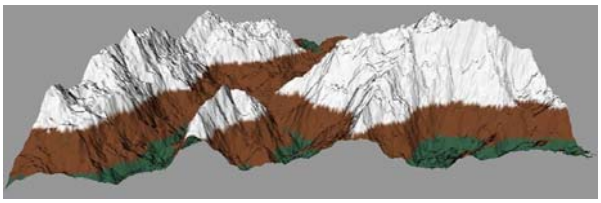
n = 64 roughness parameter = 2



n = 128 roughness parameter = 1



n = 128 roughness parameter = 2



n = 256 roughness parameter = 2

Figure 5. Simulated mountain with different *n* and roughness values with midpoint displacement method on triangle grid.

As is shown in Figure 5, the images look more natural and beautiful as the value of *n* increases.

7 Conclusion and Future Works

Fractals exist everywhere in nature and have found their way into computer graphics. They have been used for modeling the real subjects of nature too. We have shown that midpoint displacement is an efficient and simple method of generating Fractal Mountains. We can achieve *huge* bandwidth savings plus cost savings using a VRML world animated by a Java applet, rather than 2D GIF or JPEG images animated by a Java world.
It is possible to generate even more realistic and natural looking mountains than those generated by our approach. While VRML supports a very general binding model for properties (color, texture coordinate, and so on), our work can be extended in many ways. Accurate texturing, different fractal dimensions, smoothing and deformations can help to generate more realistic mountains. Visualizing Fractal Mountain by another algorithm like Diamond-Square algorithm and also visualization on hexagonal grid are interesting works, which can be followed in the future. As mentioned before, with small changes, both of our new nodes can be implemented in X3D.

Acknowledgements

We would dearly like to thank Professor Demidov's guidance on the subject and his permission to use his images in this paper.

References

BOLLE, R. M. & VEMERI, B. C. 1991. "On Three-Dimensional Surface Reconstruction Methods". *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

BRUTZMAN, D. 1998. The Virtual Reality Modeling Language and Java, *Communications of the ACM*, *vol. 41 no. 6*, June*, pp. 57-64.*

CAREY, R AND BELL, G. 1997. The Annotated VRML97 Reference Manual. http://www.best.com/□rikk/Book/ book.html

DEMIDOV, E. 2003. http://www.people.nnov.ru/fractal

FOURIER, A., FUSSELL, D. AND CARPENTER, L. 1982. Computer Rendering of Stochastic Models*, Communications of the ACM, 25, pp. 371-384.*

GeoVRML working Group. 2003. http://www.geoVRML.org.

GUEZIEC, A. AND TAUBIN, G. AND HORN, B. 1999. Framework Streaming Geometry in VRML. *IEEE Computer Graphics and Applications.*

HOWE, A. 2000. Clouds are not Spheres, Mountains are not Cones. *http://www.csc.uvic.ca/~ahowe , University of Victoria.*

JOHNSON, J. K. 1999. A Study of Generative Systems for Modeling Natural Phenomena. http://valhallawebdesign.com Thesis.

MANDELBROT, B. 1983. The Fractal Geometry of Nature, W.H. Freeman, New York.

MARTZ, P. 1996-1997. Generating Random Fractal Terrain. http://www.gameprogrammer.com/fractal.html.

MA, E. 2002. Simulating Nature. *Project.*

MILLER, G.S.P. 1986 .The Definition and Rendering of Terrain Maps, *ACM Vol 20 (4), 39-48*

NADEAU, D. 1999. Building Virtual Words with VRML, *IEEE Computer Graphics and Applications.*

PEITGEN, H. , SAUPE, D. 1988. The Science of Fractal Images*, Springer-Verlag*, New York Inc.

SALA, N. AND METZELTIN, S. AND SALA M. .2002. Applications of mathematics in the real world: Territory and landscape.

Web3D Consortium. 2003. http://www.web3d.org.