A Fractal World: Building Visually-Rich and Fully-Realistic Natural Environments

Costin A. Boiangiu, Adrian G. Morosan, and Marian Stan

***Abstract*—**Is it possible to generate a fully-developed world which looks entirely realistic, is built randomly starting from a seed and using no data whatsoever? With this paper the authors try to give an answer to this question. A set of methods from the scientific literature blended with some newly-proposed ones will be employed to generate an entire planet using only fractals, procedural models, mathematically-described objects and constructive solid geometry. Elements such as vegetation, mountains, water, waves, rocky and sandy soil and clouds will be built. Perlin noise function is used as a scalable solution for water, terrain and cloud generation. Another proposed method is for vegetation distribution, based on roulette selection, a stage of genetic algorithms.

***Keywords*—**fractal generation procedures, fractal geometry, fractal vegetation, water generation.

I. Introduction

A. History

UNCONVENTIONAL mathematician Benoit Mandelbrot created the term fractal from the Latin word "fractus", which means irregular or fragmented, in 1975. These irregular and fragmented shapes are all around us. Fractals are a visual expression of a repeating pattern or formula that starts simple and becomes progressively more complex. [1]

One of the first applications of fractals emerged long before the term had been created. Lewis Fry Richardson was an early twentieth century English mathematician who studied the length of the coast of England. He reasoned that the length of the coastline depends on the length of the measuring instrument. Measuring with a measuring instrument gives a length, but considering the more irregular coastline by measuring with a smaller apparatus gives a longer length. [2]

If there is a philosophical conclusion to reach, there is an infinite coastline containing a finite space. The same paradox was presented by Helge von Koch's snowflake. [3]

This fractal involves taking a triangle and transforming each central third of each segment in a triangular bump, in a way that makes the fractal symmetric.

Fig.1 Helge von Koch's snowflake (2008) [3]

Each bump is certainly longer than the initial segment, but still contains finite space inside. What is strange is that the perimeter moves towards infinity rather than to converge to a special number.

Mandelbrot saw this and used this example to explore the concept of fractal dimension and to prove that the measuring of the coast is an exercise of approximation. [3]

B. Characteristics

The characteristics of fractals were for the first time indicated by Alain Boutot: "It has a fine structure with details on all scales of observation; It is too irregular to be described in the language of Euclidean geometry, both locally and globally; It is a self-similar structure: it is the analog of the whole; It has fractal dimension higher than the topological dimension".

1) Self-similarity

Property of self-similarity means that the parts are similar to the whole, with variations. Fig.2 contains, on the left side, a zoomed-out image of Sierpinski Gasket, the poster child of fractals, and on the right side, there is a zoomed-in image, revealing the scaling and self-similarity the characterize fractals.[20]

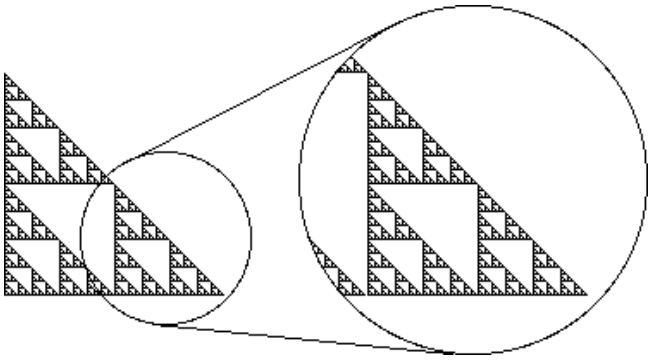


Fig. 2 Sierpinski Gasket [19]

C. A. Boiangiu is with the Computer Science Department from "Politehnica" University of Bucharest, Romania (e-mail: costin.boiangiu@cs.pub.ro). The original research carried by C. A. Boiangiu was supported by Electronic Arts Romania S.R.L.

A. G. Morosan is with the Computer Science Department from "Politehnica" University of Bucharest, Romania (e-mail: adrian.morosan@cti.pub.ro).

M. Stan is with the Computer Science Department from "Politehnica" University of Bucharest, Romania (e-mail: marian.stan@cti.pub.ro).

2) Scaling

Because of self-similarity, features at one spatial resolution are related at other spatial resolutions. The smaller features are smaller copies of the larger features. The length at finer resolution will be longer because these finer features are included. The way the measured properties depend on the resolution used to make the measurement is called the scaling relationship. [20]

3) Bounded infinity

Bounded infinity means that one can trace infinite length within a finite boundary, as demonstrated in the Koch snowflake's infinite line length circumscribing a finite area.

In other words, the fractal object can be represented by using a recursive function:

$$x, f(x), f(f(x)), \ldots \quad (1)$$

The recursive process ensures that connections will be made, but we see only parts of the whole. [20]

4) Fractal dimension

In Euclidian geometry, one dimension is represented by a line. In two dimensions, it is Cartesian space and in higher dimensions, it is a coordinate space with three or more integer number coordinates. Mandelbrot said that the fractals have the property of fractional dimensions. For a self-similar fractal, the dimension (Hausdorff dimension, named after mathematician bearing the same name) is defined as:

$$s^d = c, \quad (2)$$

where d is the Hausdorff dimension, c the amount of new copies you get after one iteration and s is the scaling factor. Using the logarithmic laws, the formula is: [22]

$$d = \frac{\log c}{\log s} \quad (3)$$

C. The elements of a fractal

A vector-base fractal is composed of two parts: the initiator and the generator. In Fig.3, the generator and the initiator for Koch Snowflake are represented.

Fig. 3 Generator and Initiator by M.M. de Ruiter [23]

It starts with an equilateral triangle as the initiator and a line that is divided into three equal segments as the generator.

The first iteration is realized by replacing every line of the initiator with the full generator. A snowflake can be approximated by iterating this operation again and again (Fig.4), replacing every line of the new initiator with the full generator. To generate a real Koch Snowflake, an infinite process is necessary, but in practice, the process ends after a finite number of iterations. [23]

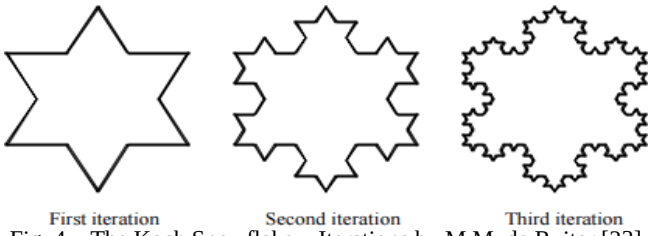


Fig. 4 The Koch Snowflake – Iterations by M.M. de Ruiter [23]

D. The generation of fractal objects

In order to generate fractal objects, several techniques are implied, all of which use the feedback processes (Fig.5) in which the output of one iteration is used as input for the next one. [24]

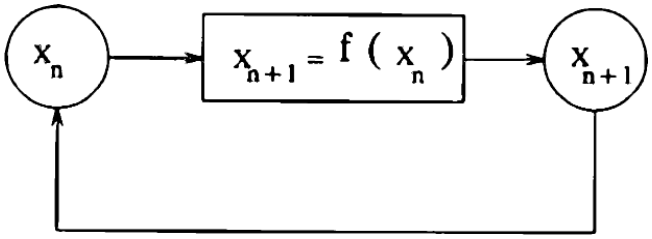


Fig. 5 Diagram of a feedback process by N. Mukaia [24]

1) Generation of fractals using formal languages

These languages, which are used for biomorphological descriptions, are known as L-systems (Lindenmayer systems). Filamentous organisms can be formalized and described by combining L-systems with branching patterns. Firstly, L-systems generate strings in a feedback loop (Fig.5). Then, in order to translate the strings into morphological description, additional drawing rules are necessary. This translation is necessary because the strings do not contain geometric information. [24].

Formal languages are the most suitable for generating plant-like objects.

2) Generation of fractal objects using Function Systems

A second approach for generating fractal objects is a method based on Iterated Function Systems (IFS). This method is used for a large class of fractal objects and it may also be applied for modelling natural objects. The IFS consists of a d-dimensional space set of mapping into itself (4) and a set of corresponding probabilities (5). [24]

$$M = \{M_1, M_2, M_3, ..., M_n\} \quad (4)$$

$$P = \{P_1, P_2, P_3, ..., P_n\} \quad (5),$$

in which: $\sum_{i=1}^{n} P_i = 1$.

3) Geometric construction of fractals

Many fractal objects may be generated with geometric constructions. The class of fractals created using this method is known as linear fractals class. A part of linear fractals can be represented by the initiator (initial polygon), generator and the production rule. In addition to the production rules geometric information is also necessary, which is stored in data structures.

The main advantage of this class of fractals is that geometric production rules can be designed interactively and in a, relatively, easy way. [24]

4) Non-linear complex mappings generation

The most important part of the fractals is represented by fractals in which relation between input and output (Fig.5) is non-linear, using complex variables and parameters. Two examples of fractals included in this class are: Mandelbrot and Julia sets. [24]

$$z_{n+1} = F(z_n) \tag{6}$$

$$F = f + ig \tag{7}$$

From (6) and (7), using decomposition the complex mapping into real and imaginary part:

$$\begin{cases} x_{n+1} = f(x_n, y_n) \\ y_{n+1} = g(x_n, y_n) \end{cases} \tag{8}$$

For each point in the complex field it can be determined how many iterations are needed until a point escapes to other points of attraction or towards infinity. For both cases, stopping criteria are formulated and for each mapping, it is possible to count how many iterations are necessary until one or more of these criteria are reached. The number iterations is used as an argument for color function. [24]

II. Related projects

The most used software programs that use concepts adjacent to this project are: Acropora, Terragen and BRL-CAD.

A. Acropora

Acropora is a procedural modeling program that combines volumetric modifiers with sampling of 3D noise of multiple octaves, to generate detailed surfaces in much less time than conventional modeling surfaces software can. Acropora adopts a less deterministic approach to generate complex and organic shapes by applying sequences of modifiers on large voxelized meshes.

The result is an infinite and continuous surface that curls and changes in a natural way. Resulted meshes contain caves, ridges, bumps and other natural features that cannot be created with height maps. [4]

Meshes generated in Acropora can be further divided into multiple segments, which can in turn be further subdivided into different levels of detail.

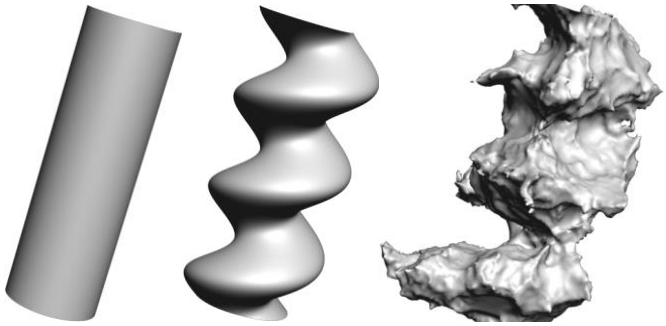


Fig. 6 Generating a noise spiral in Acropora (2015) [4]

1) How it works

A mesh generation begins with a basic geometry and it combines the complex noise sampling with the user-specified form modifiers to generate a single field density, which is then marked to generate a surface that becomes the final mesh.

After the isosurface is extracted, then the user can level, optimize or filter the resulting mesh. Evaluation of the density function is numerically intensive, therefore, the volume is often broken into blocks. However, mesh-sized operations can be performed globally on the entire mesh or on the selected blocks. Acropora automatically takes care of alignment, ensuring a joint line between the adjacent mesh sites.

GPU support offers real-time response when editing volumes. In addition to applying the global modifiers, the user can create localized regions or volumes, which work in tandem or independently of other regions.

Acropora can also be used to test and develop procedural surfaces that later can be used for generating meshes in hardware. [4]

2) Semi-deterministic approach

Math-Modeling of voxels is by its nature a non-precision approach to sculpting surfaces. While 3D models can be sampled by voxels, Acropora renders complex surfaces that are difficult to render using conventional modeling.

The complex surfaces that can be rendered in Acropora are caves, complex organic shapes, and realistic clusters of rocks, projections and canyons. The user can control the intensity and quality of the surface structures.

B. Terragen

Terragen is a program for Microsoft Windows and Mac OS X, developed and published by Planetside Software, which generates landscapes. It can be used to create renderings and animations of landscapes. [5]

Terragen is a designing and rendering software for photorealistic 3D environments. It is widely used in a broad spectrum of industries from visual effects to games or art. [5]

1) Overview

Terragen is popular among amateur artists, because it has a simple freeware version with an intuitive interface and it is able to create photorealistic landscapes. It can also use DEM

files and other surface maps for rendering.

The commercial version of the software available is able to create larger fields, renderings with higher resolutions, use larger terrain files and use a better anti-aliasing algorithm.

The terrain, clouds and objects are constructed from two dimensional height maps that are procedurally generated.

Terragen is not a game engine, it uses algorithms that simulate skies, outdoor lighting, and terrain textures, being also capable of rendering very large and very detailed terrains. [5]

C. BRL-CAD

BRL-CAD is a powerful modeling combinatorial system, which is cross-platform and open source. It includes interactive 3D geometry editing, ray-tracing support for high quality rendering and geometric analysis, support for network distributed framebuffers, tools for image and signal processing, path-tracing support and photon mapping for realistic image synthesis, a suite of performance evaluation and analysis systems, embedded scripting interface libraries for robust geometric representation and high performance analysis.

For more than two decades, BRL-CAD was a solid modeling CAD system used primary for the US military to model weapons systems for vulnerability analysis and lethality [6]. This package was also used in the planning of radiation doses, medical views, computer graphics education, CSG concepts and modeling education, performance evaluation tests of systems and many other purposes. Today the solid modeling system is often used in a wide range of applications, from academic to industrial and military, including the design and analysis of vehicles and mechanical parts.

BRL-CAD supports a wide variety of geometric representations including an extensive set of traditional CSG primitive solids such as boxes, ellipses, cones and tori, as well as explicit solids made from closed collections of B-Spline uniform surfaces, non-uniform rational B-Spline, n-Manifold geometry and mesh geometry.

All geometric objects that are obtained in BRL-CAD can then be combined using Boolean CSG operations, including unions, intersections and differences. [6]

III. Algorithms

After presenting the fundamental theory and existing applications, the paper will now go into more detail and present ways for building objects like water waves, lands with mountains, clouds, vegetation, soil with sand or rocks and land textures, generated mathematically. [7]

A. Generation of water

To generate water waves there are several algorithms. The main types of waves used are sine waves, Gerstner waves and FFT waves. There are also algorithms for generating turbulent waves, algorithms for generating waves for shallow water and algorithms for generating movement of incompressible viscous fluids using the Navier Stokes equations.

Everything can be calculated mathematically, including water color and its reflection and refraction.

There are also developed algorithms for generating caustic and foam, water drops and bubbles, using systems of particles. [7]

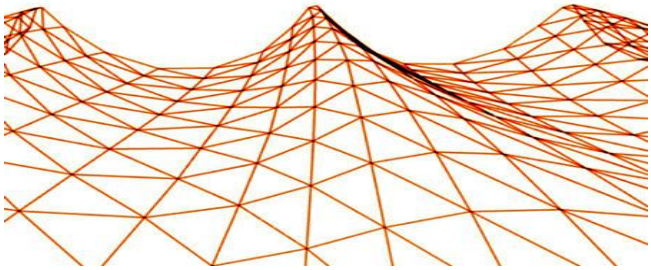


Fig. 7 Gerstner Waves (2007) [7]

1) Sine Waves

Sine waves are calculated using sums of sines, which are continuous functions that describes the height and orientation of the water surface at all points of the plane.

The height of each wave according to the horizontal position (x,y) and time t is defined as:

$$W_i(x,y,t) = A_i \cdot \sin\left(D_i \cdot (x,y) \cdot w_i + t \cdot \varphi_i\right) \quad (9)$$

where:

A_i is the amplitude,

D_i is the direction, the horizontal vector that is perpendicular to the wave front which moves along the top of the wave,

w_i is the frequency,

φ_i is the phase constant and

$\varphi = S \cdot \omega$, where S is the speed, also known as the distance the wave travels at the top of a frame, and $\omega = 2\pi / L$, where L is the length of the wave, that is the distance between the peaks of the waves.

For all (x,y) in the horizontal plane 2D, the 3D position of the sine waves surface is:

$$P(x,y,t) = (x,y,H(x,y,t)) \quad (10)$$

where $H(x,y,t)$ is the total area, and is defined as:

$$H(x,y,t) = \sum W_i(x,y,t).$$ [7]

2) Gerstner waves

Unlike sine waves, which have a rounded appearance and are suitable for calmer waters, Gerstner waves can control steepness, thus sharp waves can be generated, making them more suitable for rough waters.

For all (x,y) in the horizontal plane 2D, the 3D position of the Gerstner surface waves is:

$$P(x,y,t)=\left(x+\frac{\partial(H'(x,y,t))}{\partial x},y+\frac{\partial(H'(x,y,t))}{\partial y},H(x,y,t)\right) \quad (11)$$

where t is the time,

$\frac{\partial(H'(x,y,t))}{\partial x}$ is the partial derivative of $H'(x,y,t)$ in x direction and is defined as:

$$\frac{\partial(H'(x,y,t))}{\partial x}=\sum(Q_iA_i\times D_i.x\times\cos(w_iD_i\cdot(x,y)+\varphi_it))$$

$\frac{\partial(H'(x,y,t))}{\partial y}$ is a partial derivate of $H'(x,y,t)$ in y direction and is defined as:

$$\frac{\partial(H'(x,y,t))}{\partial y}=\sum(Q_iA_i\times D_i.y\times\cos(w_iD_i\cdot(x,y)+\varphi_it))$$

$H'(x,y,t)=H(x,y,t)\cdot Q_i/w_i$, $H(x,y,t)$,W_i, D_i, A_i and φ_i are the same as in the case of sine waves.

In addition to the sine waves, Gerstner waves control the inclination of the wave through the Q_i parameter. If its value is zero the obtained Gerstner waves are similar to sine waves.

Unlike sine waves, the Gerstner waves and points move sideways and even if the overall wave height is the same, Gerstner waves are more realistic because sharp waves can be obtained. [7]

3) FFT waves

FFT waves are not based on any physical model, but they are based on statistical models emerged from actual observations of oceans and seas. These waves have been used commercially several times, especially for movie animation seas and oceans.

In statistical models and crafts, wave height is a random variable, being a function of horizontal position and time. In order to obtain FFT waves the wave height field is split into a set of sine waves with different amplitudes and phases, after which inverse FFT transform is used to rapidly assess the amount obtained. [7]

4) Agitated waves

The FFT algorithm for generating waves produces waves that have rounded edges, suggesting calmer weather. To get restless waves, with sharper peaks and flatted bases, instead of directly modifying the height field, the positions of the points need to be moved horizontally.

FFT waves are an alternative for sine waves, while the restless waves are an alternative Gerstner waves, and although the algorithms to generate the waves are different, the results are similar. [7]

Another method to simulate agitated water is by using Perlin noise. The advantage of scalability makes Perlin noise the most suitable option to simulate very wide surfaces. (Fig. 8)

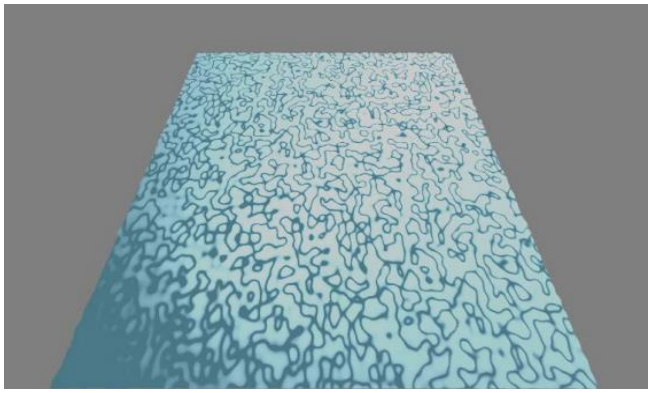


Fig. 8 Perlin noise water

5) Waves for shallow waters

For shallow water, waves are generated with the Saint Venant equations that are partial hyperbolic differential equations which describe the flow of a fluid underneath a surface of pressure. [7]

6) Fluid Movement

Incompressible viscous fluid motion is described entirely by the Navier Stokes equations. In these equations there are three types of forces that act: body forces, pressure and viscous forces.

Body forces are the forces acting on the entire surface of the water. It is generally assumed that these forces are formed only from gravity. Pressure forces act upon the inside of the fluid and upon the surface normal, and the forces due to friction of the water are the viscous forces and they act in all directions over the entire surface of the water. [8]

7) Water Color

The color of the water is generally given by the reflection and refraction, but the water may also have a color which depends on the direction of the incident light beam, the direction of viewing and the properties of the water. [8]

8) Reflection and refraction of water

Most visual effects of the water are caused by the reflection and refraction. This occurs when a ray strikes the surface of the water, and part of it is reflected back into the atmosphere and the other part is refracted in the volume of water.

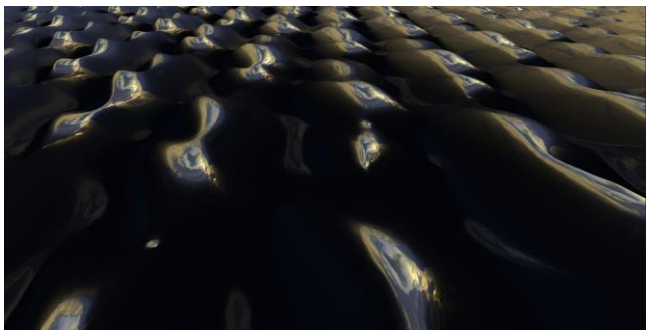


Fig. 9 Ocean with Gerstner waves, reflection and refraction by Jerry Tessendorf (2005) [8]

One of the most important visual aspects of realistic rendering of water is obtained through the Fresnel equation

which defines a factor of combination between reflection and refraction. [8]

Fig. 9 presents the current state of the project that includes Gerstner wave generated water, and water color formed by reflection and refraction.

9) Caustics

Caustics result from reflected or refracted light rays on a curved surface and therefore they focus only in certain areas of the receiving surface. They can be generated "fractally" or mathematically. [9]

10) Foam, drops and water bubbles

When the water surface is very agitated or when it encounters obstacles it generates foam, water drops and bubbles due to breaking waves. This can be achieved by means of a system of particles that will be based on Newtonian dynamics. [9]

11) Generation of rivers

The classical method for generating rivers is a part of height-map generation algorithm. The algorithm generates a terrain model around a precomputed set of ridge lines and rivers network. First, a rapid method that generates the ridges and rivers network is used. Then, an extension of the basic midpoint displacement method is applied for generating fractal terrain model around a pre-filled ridges and rivers network. [18]

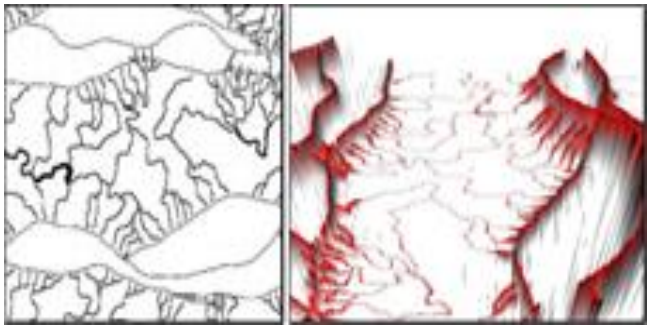


Fig. 10 The Midpoint Displacement's Inverse process by F. Belhadj (2005) [18]

In Fig.10, the left image shows: in black color the computed elevations and in the white color the elevations that still have to be computed. In the right image, in black/gray color, the tridimensional view of D.E.M is represented, in red, the elevations computed with the M.D.I. process and in white, the elevations that still have to be computed. [18]

The second approach is a post-processing step on the existing height-map. For the first one, a generated river network forms a basis from which a height-map is inferred. For the last one, a height-map is analyzed to find the potential stream routes from mountains into valleys. [15]

12) Generation of oceans and lakes

Procedural water bodies, such as oceans and lakes and their connections, stream networks, deltas and waterfalls are similarly generated. Oceans are commonly generated by setting a fixed water level and, for lakes, the classical method is a flooding algorithm from points of low elevation. [15]

B. Generation of land

For the generation of fields with mountains there are multiple algorithms. [10]

1) Transcendental land

Transcendental lands are lands mathematically calculated using sine and cosine functions. Transcendental lands have a rounded appearance and relate to sine waves resembling water. [10]

2) Fractal land

Fractal terrains are more realistic than the transcendental terrains, due to the sharp mountain peaks. These fields can be generated using several algorithms, among which the most important are the median shift algorithm, Diamond-Square algorithm and FFT. [10]

3) Median shift algorithm

Median shift algorithm provides realistic results. This algorithm assumes that the start is a line between two points, then the median of that line is moved with a random value in the vertical direction, then the medians of the two new segments are moved with random values in the vertical direction and then the previous step is repeated until the desired level of detail is reached. [11]

4) Diamond-Square algorithm

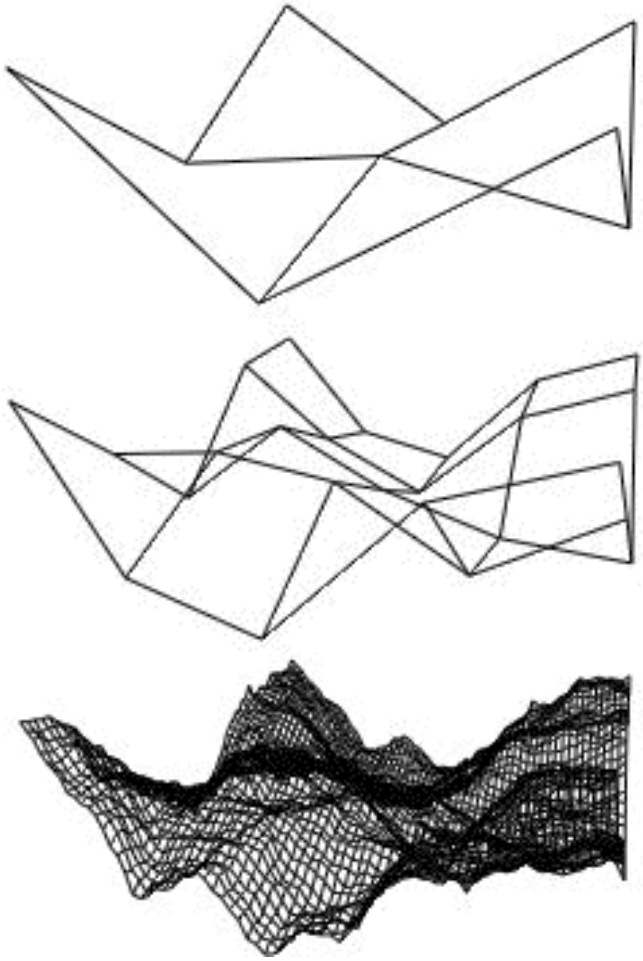


Fig. 11 Fractal terrain generation using Diamond-Square algorithm by David P. Feldman (2012) [11]

The Diamond-Square algorithm generates more realistic terrain than the previous algorithm, because the latter one leaves square objects in the field. The Diamond-Square algorithm mitigates this by alternating calculated values of the middle points of squares and diamonds. This algorithm assumes the start by assigning a random height of the four corners of the grid, and then averaging four corners, plus a random disturbance and assigns this value to the middle point of the square formed by four points. Then it takes each achieved diamond, it calculates the average of the four corners of each diamond, plus a random disturbance and assigns the middle point of the diamond. The previous two steps are then repeated until reaching the desired level of detail. [11]

Fig.11 shows the generation of a fractal terrain using Diamond-Square algorithm, from the initial stages of low level detail in the final stages with high level of detail. [11]

5) Improved Diamond-Square Algorithm

In order to accelerate the process of generating terrain, it is used an improved algorithm. The goal of both algorithms is to construct a bidimensional heightmap. In the standard algorithm the 2D array is a square (Fig 12.a). The mid-point value is the average of the angles plus a random value, which is reduced over the steps.

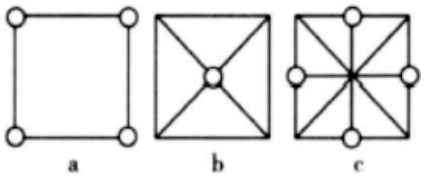


Fig. 12 The improved Diamond-Square algorithm [25]

The standard algorithm divides the square into four triangles (Fig 12.b). In addition, the improved algorithm draws two lines between the mid-point and the mid-points of hypotenuses of the triangles, dividing each triangle into two small triangles. (Fig 12.c) [25]

6) FFT algorithm

Unlike the movement of the median algorithm and Diamond-Square algorithm which have linear running time, the FFT algorithm's runtime is logarithmic.

Another difference of the FFT algorithm is that the terrain has a more rounded texture and has no raised or pointed peaks. [12]

At distance either less frequencies can be considered, or the terrain simplified for speed purposes. [14]

7) Erosion fractal

The height-maps can be transformed using simulations of physical phenomena, such as erosion. In order to diminish the sharp changes in elevation, thermal erosion is used, by iteratively distributing material from higher to lower points, until the maximum angle of stability for a material is reached.

Another type of erosion is caused by rainfall or fluvial erosion. This type can be simulated using cellular automaton (model of a system of "cell" objects), where dissolved material that flows out to other cells and the amount of water are calculated based on the local slope of the elevation profile. [15]

8) Digital elevation models (DEM) method

Besides other methods, this setup allows the user to interactively edit the height-map. A user draws a 2D map of polygonal regions, each of which is marked to have a certain elevation profile. The straight boundaries of the regions are perturbed and rasterized in a grid. Then, for each region, DEM data is selected to match the requested elevation profile. The selection of data is realized using genetic algorithms.

The main advantages of this method are: realism, extensibility and ease of use (intuitive control, with low input requirements). However, the generated transitions at the boundaries between regions are still rather abrupt. [16]

C. Cloud Generation

To generate clouds, noise functions are used in general. The noise that offers the most realistic results is the Perlin noise. Meteorological phenomena, such as lightning, can be mathematically generated or through fractals. Rain and snow, can be generated using a particle system. [7]

Another method to generate a cloud data model is based on the random displacement algorithm. The process is similar to the generated terrain. The main difference is that we need a height map where vertexes have different colors and not different heights.

To accomplish this, there is an additional final step in which the heights are mapped to colors. Firstly, the basic clouds are generated through the Diamond-Square algorithm. Based on the height map, we figure out the minimum and maximum height. These two values determine the mapping interval for heights to color interval. [25]

1) Noise functions

The role of noise functions is to provide a pseudo-random signal, efficiently implemented and repeatable over a three-dimensional space. In general, the sound functions receive an integer as a seed and they return a pseudo random number based on the received parameter. [7]

2) Perlin Noise

To create a Perlin noise function, a noise and an interpolation function are required. The noise function must be, in general, a random number generator that will return the same value if two numbers with the same value are sent as input and different values if two different values are sent as a parameter.

A standard interpolation function receives three parameters, two of which represent the values between the return value that must be interpolated and the third parameter based on which the returned value is interpolated. [13]

D. Generation of vegetation

Vegetation can also be generated with fractals or mathematically. Procedural vegetation is a classic research topic in the field of procedural modelling and includes both procedures for generating 3D plant models and trees and specific methods for placement on a given surface. [7]

1) Generating plants

There are many plants that can be generated by fractals or mathematically, including ferns, grass, flowers, fruits, vegetables or leaves. Plants can be generated using L-systems

or using iterated function systems. [7]

Branching patterns may be defined as L-systems by using K denoted by <G, W, P>, in which G is a set of symbols, W is the starting string and P is the production rule.

Monopodial branching, in which the growth of the main axis continues throughout the plant's life, may be represented by the following L-system [24]:

$$KM = \langle G, WM, PM \rangle \tag{12}$$

$$G = \{0,1,[,]\} \tag{13}$$

$$WM = 0 \tag{14}$$

$$PM = \{0 \to 1[0]0, 1 \to 1, [\to [,] \to]\} \tag{15}$$

<Iteration>	<Generated String>
1:	0
2:	1[0]0
3:	1[1[0]0]1[0]0
4:	1[1[1[0]1]1[0]0]1[1[0]0]1[0]0

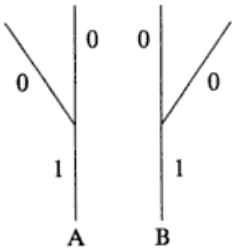


Fig. 13 Drawing rule for the string 1[0]0, shape A and B by N. Mukaia [24]

The drawing rule showed in Fig.13 is one possible way to visualize the generated string.

An alternative system to procedurally model plants is by placing plant components in a graph. Connected components can be structured in sub-graphs. The system traverses this graph, generating and placing instances of the components in an intermediate graph that is used for geometry generation.

A set of components is connected by the user to describe the structure of the plant. The algorithms are controlled by graphical user interface on the basis of spline functions. [17]

2) Generation of trees

Trees can be generated using L-systems or iterated function systems. An L-system is a classical and often used example of a rewriting system. Although the L-systems are used for rewriting strings of text, the resulting set of symbols can be interpreted in 2D and 3D. One of the best known such tree is the tree of Pythagoras, which is built recursively. [7]

The L-system has an advantage in describing the random growth of leaves, flowers, fruits, freely and easily, but it is not suitable for textures, which also have self-similarity. In order to solve this problem, the iterated function system (IFS) is used. The main advantage of the IFS is that the graph is a colored map, texture mapping being realized easily. On the other hand, the plants are very similar. A combination of these two methods gives the best results.

The leaves are simulated via IFS and the branches are generated via L-systems. The idea of these two methods is to generate the tree using L-systems that control the IFS's code parameters. The most important advantages are: implementation is easy, trees look realistic and the shape detail of the leaf is more attractive. [27]

Without any random affection, the tree seems too stiff and similar. Trees are affected by the wind and gravity and in order to create more natural trees, random functions are used. The random values are used to disturb the branch's dip angle, length and size. Fig 14 (a) depicts a tree without any random values and Fig 14 (b) a tree generated by L-systems controlling IFS's code parameters, but with added randomness.

Fig. 14 Trees generated by L-system controlling IFS code's parameters: (a) without random (b) with random

3) Vegetation distribution

Vegetation distribution is provided by two approaches: explicit specification and procedural generation. The first one is obtained by surveying a forest or specified by the user. The second is realized using a point pattern generation model or an individual-based population model. Both methods use detailed information in order to construct an ecosystem.

The problem is that these methods could not offer a suitable option for small areas.

Because random selection does not offer a natural distribution, we came up with a method, inspired from genetic algorithms, more precisely, the roulette selection.

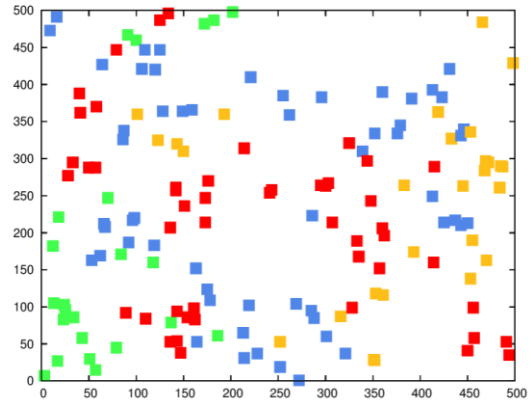
Roulette selection is a method to pick the most suitable parents that are used to create the next generation. Each parent has assigned a selection probability based on a fitness value. A fitness function is an objective function which describes the figure of merit, how close the solution to achieve a set of goals is.

In our case, we just need to decide what species is assigned to a tree. Firstly, a sample from each species is planted. In order to avoid plantation of seeds in water or in undesirable places, previous checks are required. After this step, for each element we want to plant, we need to decide which species is the closest, by calculating the Euclidian distance, or any other distance, between the element and other trees that are already planted.

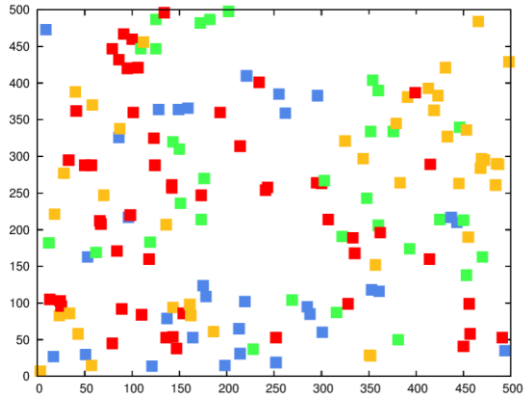
The fitness is inversely proportional to the minimum distance. We want a higher fitness values for the closest species. For fitness function, we selected the function:

$$fitness(x) = \left(\frac{1}{dist(x)} \right)^n,$$

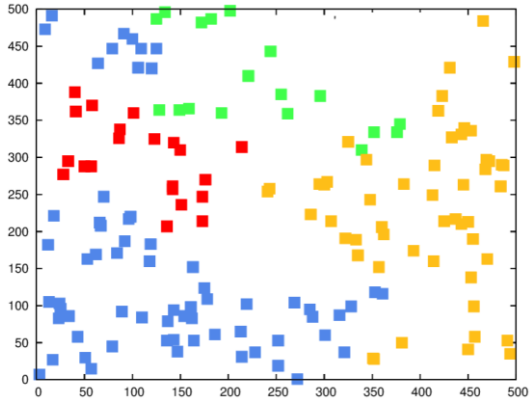
where x is the species and $n \in [-1, \infty)$.



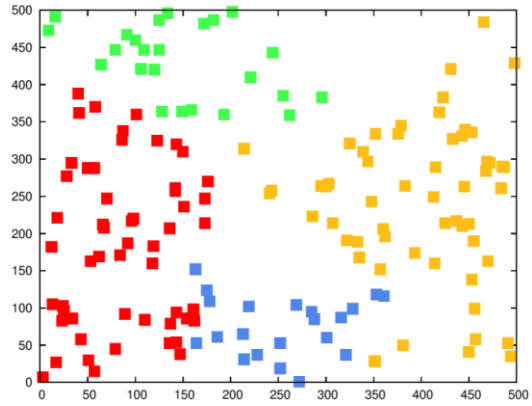
(a) n=-1



(b) n=-2



(c) n=-4



(d) n=-6

Fig. 15 Vegetation distribution

In small regions n must be lower, because the minimum distances are similar and the entire ecosystem looks random (Fig 15-a). When the n value becomes lower, the entire distribution tends to concentrate in clusters.

On the other hand, a very organized ecosystem like in Fig 15-d does not look natural, either. In conclusion, the n value must be chosen in correspondence with the dimensions of the region, for natural results.

After the fitness for each species is computed, we need to calculate the selection probability for each species. The function used for calculating the probability is:

$$P(x_k) = \frac{fitness(x_k)}{\sum_{i=1}^{n} fitness(x_i)},$$

where n is is the number of species and x_k is the species for which the selection probability is calculated.

Because the sum of all probabilities is equal to 1, in the final step we distribute the probability ranges in the subunit interval [0, 1]. After we have the interval, we need a random subunitary number, based on which the selected species is decided.

This method is the most suitable when we have small regions with no detailed information about the ecosystem or it is impossible to apply altitudinal vegetation zones.

E. Soil generation

The most important elements of this category that can be generated are rocks and sand. Sand dunes using noise functions can also be generated. [7]

A. Peytavie1 and E. Gali came up with a complex framework to simulate the material layers. A terrain is defined as two dimensional grid of material stacks. (Fig. 16)

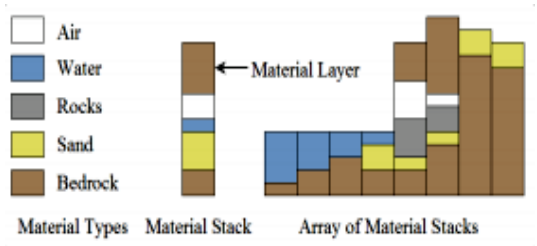


Fig. 16 Material stacks[28]

A material stack contains different material layers that are characterized by their thickness and the corresponding material type. The types of material implemented in the framework are: air, water, sand, bedrock and rocks.

Rocks are created using Voronoi cells. The first step is to generate a cubic tile containing these cells and the next step is the erosion. The process of erosion is realized at some random contact points so that the rocks should nicely pack together.

The rock instantiation is realized based on the material layer. The instantiated rocks like their Voronoi center within a

rock material layer [28]

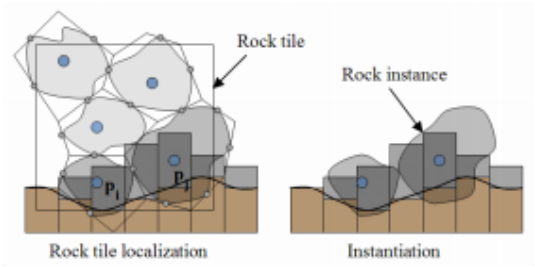


Fig. 17 Rock instantiation process [28]

The elimination of small rocks that are not visible is a method of optimization. Therefore, all rocks whose distance to the surface is larger than twice the maximum radius of Voronoi embedding spheres are eliminated.

Another important aspect of soil generation is the addition of granular material. The approach is a three dimensional generalization of painting tools for editing height fields. [28]

As we can see in Fig. 18, a brush is characterized by a depositing region and the distribution material. This method enables the user to control the amount and the location of granular material.

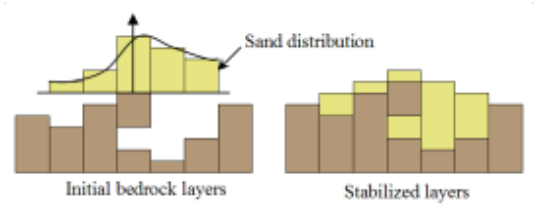


Fig. 18 Granular material on initial bedrock layer [28]

F. *Urban environments*

The common approach for procedurally generating cities is to start from a dense road network and identify the polygonal regions enclosed by streets. Building lots are the result of subdivision of these regions. Then, for populating these lots, the lot shape is used directly as the footprint of a building (Fig.19).

Fig. 19 Procedural sketching session: a) river flowing towards the sea, b) road feature crossing the river, c) city created along the river banks by R.M. Smelik (2011) [15]

Another method is to fit the building footprint on the lot. By simply extruding the footprint to a random height, a city of skyscrapers and office buildings can be generated. These approaches are used to create a macro environment; for more complex details, several rule-based methods are necessary. [15]

G. *Generation of microorganisms*

One direction of application of fractals in biology is to artificially create biological objects or systems. A new concept of Mandelbrot was demonstrated by C.A. Pickover by coloring the created images closely corresponding to single cellular organisms. These organisms can be seen in Fig 20, and they are also named biomorphs. [26]

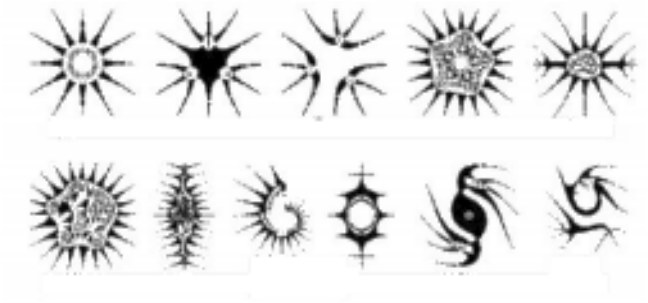


Fig. 20 Examples of cellular organisms created using fractals [26]

H. *Generation of textures*

Textures can be generated using height maps obtained at the terrain generation. [7] Another method is used for generating texture image such as surfaces of houses or roads. For this, random or fractal functions are used. This method uses two types of images. One is a material image and the other is a weathered one. The material image is generated by placing some fundamental patterns at random. The more patterns are used, the more varieties are generated. On the other hand, the weathered image is generated using fractal functions.

Intended images, such as a texture that represents a water drop track on the wall, can be realized by selecting the fractal seeds. Also, combining both material and weathered images can make more realistic images for texture mapping. The generated images look very natural and enable the very realistic data modelling of landscape simulation. [24]

Another method for texture generation is using Perlin noise. Because one of the most important advantages is the scalability, Perlin noise is suitable for texturing large regions of terrain.

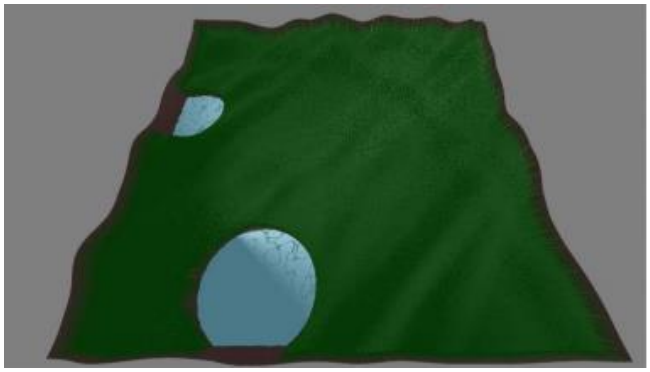


Fig. 21 Large-scale Perlin noise for normal map distortion

In Fig. 21, Perlin noise was generated at a very large scale to simulate grass, using normal map distortion.

The same function, generated at a small scale was used to simulate different levels of aridity, by changing the base color as in Fig. 22.

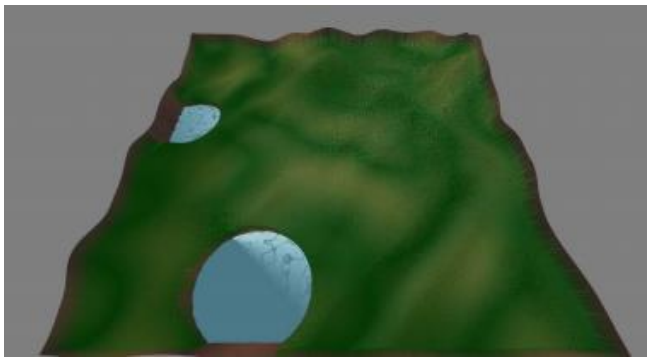


Fig. 22 Small-scale Perlin noise for levels of aridity

I. Generation of other elements

The most important of these elements are animals such as spiders, snails, peacocks, sea urchins and starfish, but the generation of stalagmites and stalactites and crystals is also possible. [7]

IV. Conclusions

This paper presented techniques to generate many classes of objects on the basis of equations, possibly with minimal data, such as a seed. Items that can be randomly generated include: terrain, terrain textures, terrain elements (sand, rocks), clouds, vegetation, soil and water. All the aforementioned generation techniques were included in "A Fractal World" project designed by the authors to prove that simple mathematical techniques may be employed to generate realistically environments from virtually no data and randomly evolving starting from a seed and running up to a custom level of detail. The project was designed with educational purpose in mind, carrying the strong desire that in the end it will enclose almost all the important ideas in the fractal world generator and also it will allow students to customize the techniques, tweak the algorithms for even more realism and perform optimizations for better viewing performance. "A Fractal World" [29] aims at generating "fully-mathematical" planets using a configurable amount of detail and computing power and to serve as a powerful educational tool for computer graphics students to integrate their knowledge and original ideas.

Acknowledgment

The authors want to thank Iacob Stefan Octavian and Mihai Zaharescu for their help with text management and corrections.

References

[1] B. Mandelbrot, *The Fractalist: Memoir of a Scientific Maverick*, Knopf Doubleday Publishing Group, 2012.
[2] L. F. Richardson, O. M. Ashford, P. G. Drazin, *The Collected Papers of Lewis Fry Richardson*, Cambridge University Press, 2009
[3] NOVA, *Hunting the Hidden Dimension*, PBS, 2008
[4] http:// www.voxelogic.com/ , "*Welcome to the Voxelogic*", Accessed on: 2/1/2015
[5] http:// planetside.co.uk/ , "*Terragen 3*", Accessed on: 2/1/2015
[6] http:// brlcad.org/ , "*BRL-CAD | Open Source Solid Modeling*", Accessed on: 2/1/2015
[7] NVIDIA, *GPU Gems*, NVIDIA Corporation, 2007
[8] J. Tessendorf, *Simulating Ocean Water*, 2005
[9] Y. Hu, L. Velho, X. Tong, B. Guo, H. Shum, *Realistic, Real–Time Rendering of Ocean Waves*, 2015
[10] K. Bird, T. Dickerson, J. George, *Techniques for Fractal Terrain Generation, HRUMC*, 2013
[11] D. P. Feldman, *Chaos and Fractals: An Elementary Introduction*, Oxford University Press, 2012
[12] M. F. Worboys, M. Duckham, *GIS: A Computing Perspective, Second Edition*, CRC Press, 2004
[13] D. S. Ebert, *Texturing & Modeling: A Procedural Approach*, 2003
[14] C. A. Boiangiu, B. Raducanu. "*3D Mesh Simplification Techniques for Image-Page Clusters Detection". WSEAS Transactions on Information Science, Applications, Issue 7, Volume 5*, pp. 1200 – 1209, July 2008
[15] R.M. Smelik. "*A Declarative Approach to Procedural Generation of Virtual Worlds*" , 30 november 2011
[16] [16] R. L. Saunders. "*Terrainosaurus: Realistic Terrain Synthesis Using Genetic Algorithms*", December 2006
[17] O. Deussen, C. Colditz, L. Cocunu, H.C. Hege. "*Efficient modelling and rendering of synthetic landscapes*"
[18] F. Belhadj, P. Audibert. "*Modeling Landscapes with Ridges and Rivers*", November 2005
[19] https:// classes.yale.edu/ fractals/ IntroToFrac/ SelfSim/ SelfSim.html, "*Self-Similarity*", Accessed on: 2/1/2015
[20] L. Strudwick. "*Infinite Space and Self-Similar Form in Alchemy and Fractal Geometry*"
[21] A. Shamsgovara. "*Analytic and Numerical Calculations of Fractal Dimensions*". Research Academy for Young Scientists, July 11, 2012
[22] M. M. Ibrahim and R. J. Krawczyk. "*Generating Fractals Based on Spatial Organizations*". College of Architecture, Chicago, IL USA
[23] M.M. de Ruiter (Ed.). "*Advance in Computer Graphics III*". EurographicSeminars
[24] N. Mukaia, Y. Sakaguchia, M. Kosugia. "*A Method for Generating Texture Images used on Landscape Simulation*". Electronic & Computer Engineering, Musashi Institute of Technology, Tokyo, Japan
[25] D. Fang. "*The Study of Terrain Simulation Based on Fractal*". WSEAS Transactions on Computers. College of Computer and Information Zhejing Wanli University, Ningbo, 315100
[26] R. Jovanovic, M. Tuba. "*A Visual Analysis of Calculation-Paths of the Mandelbrot Set*". WSEAS Transactions on Computers. Institute of Physics, Belgrade, Pregrevica 118, Zemun, Serbia. Faculty of Computer Science, Megatrend University of Belgrade, Bulevar umetnosti 29, N. Belgrad, Serbia.
[27] D. Fang, Xi Li-Fneg. "*An Application of L-system and IFS in 3D Fractal Simulation*". WSEAS Transactions on Systems. College of Computer and Information, Zhejing Wanli, University Ningbo, China.
[28] A. Peytavie, E. Galin, J. Grosjean, S. Merillou. "*Arches: a Framework for Modeling Complex Terrains*". EUROGRAPHICS 2009 / P. Dutré and M. Stamminger. LIRIS - CNRS - Université Claude Bernard Lyon 1, France.
[29] C. A. Boiangiu, A. G. Morosan, M. Stan, "*Fractal Objects in Computer Graphics*", Proceedings of the 6th International Conference on Applied Informatics and Computing Theory (AICT '15), Salerno, Italy, June 27-29, 2015, WSEAS Press, pp. 123-131.