HARDNESS DATA SYNTHESIS FOR HEIGHT-FIELD BASED LANDSCAPE MODELS

Korneliusz WARSZAWSKI[1], Slawomir NIKIEL[2], Tomasz ZAWADZKI[1]

[1]Faculty of Electrical Engineering, Computer Science and Telecommunications,
University of Zielona Gora, ul. Podgorna 50, 65-246 Zielona Gora, Poland

k.warszawski@weit.uz.zgora.pl
t.zawadzki@weit.uz.zgora.pl

[2]Institute of Control and Computation Engineering,
University of Zielona Gora, ul. Podgorna 50, 65-246 Zielona Gora, Poland

s.nikiel@issi.uz.zgora.pl

Abstract

Landscape modelling is a key element of synthetic environments in virtual reality systems such as: military training systems, virtual reconstruction of cultural heritage, digital entertainment and game development. Most popular and simplest to implement data structures for terrain models are based on the height-field representation. Unfortunately, such models do not contain information about hardness of terrain or it susceptibility for erosion forces in different landscape areas. It is possible to derive hardness data from actual geological records, but for virtual environments we propose the fully automated synthesis of hardness information for terrain models. Our method stems from classical Poisson Faulting algorithm, that was originally used by Mandelbrot and Voss to model ragged landscapes and whole planets. The proposed technique can help to generate hardness data under the height-field base terrain, with regard to different geological materials. The algorithm ensures that those materials are not randomly displaced but form clusters throughout the entire virtual terrain. In addition, the technique can be simply parallelized and implemented in CUDA programming environment. Obtained data structure can be used for further synthesis of eroded landscapes or can work as a test bed for comparison of different geological erosion simulating some models.

Keywords: Hardness generation, landscape modelling, erosion model, virtual reality

INTRODUCTION

Remarkable forms of virtual terrains are possible when developers, artists and virtual world builders spend a lot of time manually deforming polygon meshes. Alternatively, an acceptable visual level of recognized geological structures can be obtained much faster by some automated methods of terrain modelling. We can observe constant development e.g. in chaotic and dynamical systems that may open some paths to modern automated modelling procedures (CLEMPNER and POZNYAK (2011), DI TRAPANI and INANC (2010)).

Applications of those techniques have been used as elements of virtual environments in military and civilian training courses, simulations of cultural heritage reconstructions, digital entertainment, game development and virtual reality systems (BOHEMIA INTERACTIVE SIMULATIONS (2011), BONK and DENNEN (2005), RICKEL and JOHNSON (1999), SMELIK et al. (2010), WELLS and DARKEN (2005)).

Landscapes described as voxel-based model *(voxel maps)* enable modelling terrain shapes and their internal structure, e.g. caves. Such models offer the most accurate representation for complex terrain structures, however the major inconvenience is their complexity and 'greed' for memory storage and computing resources. For most models that are used in simulations, where internal structure is not important case, a classical height-field *(height map)* representation is sufficient. Such a model contains only information about elevation of terrain nodes at coordinates defined by index of row and column of this data structure (MUSGRAVE (1993), LENGYEL (2010)).

This kind of data is a challenge when we want to simulate erosion processes. Implementation of hardness data generation technique, may offer effective solution for this types of inconveniences.

TERRAIN MODELLING FOR VIRTUAL LANDSCAPES

The classic model

The height-field representation is the most popular structure for landscape shape description. Major advantage of such models is easy implementation of the underlying data structure and relatively simple algorithms for rendering, Level Of Detail *(LOD)*, texturing and shading. The discreet mathematical model is organized as *m-by-n* matrix where cells stores records about elevation at position indicated by indexes of row and column. This kind of model is suitable for description of landscape surface without complex elements e.g. rock shelves, caves, cliffs, etc. (FRADE et al. (2009), MANDELBROT (1982), MUSGRAVE et al. (1989), MUSGRAVE (1993), LENGYEL (2010)).

The voxel model

Limitations of landscape models based on height-field structure result from planar space representation. It makes them unacceptable for high visual realism. Any structure similar to caves, cliffs, rock shelves or any vertical surface cannot be described by that model. This inconvenience can be solved with defining landscape as scalar function over three-dimensional grid *(voxel map)*. Major weakness of the voxel-based method is that it consumes lots of system memory resources. Also the rendering process needs complex algorithms with less performance than in the classical approach (FRADE et al. (2009), LENGYEL (2010), WAN et al. (1999)).

The LDR model

Benes and Forsbach proposed a Layered Data Representation *(LDR)* of terrain model, that was a compromise between the methods of height-field and voxel landscape description. Similar to the classic approach, the discreet mathematical model is organized as k-layered m-by-n matrixes, where cells contains information about elevation of the layer and other attributes e.g. density, water or gas depositions. Total height of terrain at given coordinates is a sum of heights at each layers on the corresponding positions. In modelling process, those authors assumed that zero-height layer do not exist in general model, which results in increased performance of the algorithm. In addition, when cells in the layers contains zero-density or include water or gas, then those cells represent caves or holes in the terrain structure. This property is a major advantage over the classic height-field depiction of landscape models and is similar to the voxel representation (BENES and FORSBACH (2001), BENES and FORSBACH (2002), BENES (2007)).

TERRAIN MODEL WITH HARDNESS DATA

In our research, we assumed that the mathematical models for virtual landscapes are based on two-layered representation. Both layers are discrete models organized as *m-by-n* matrix, where *m, n* $\in N$.

The height-field layer

The first layer *(H)* represents a classic height-field and is a function defined in equation 1, where each cell represents the elevation value at coordinates defined by indexes of a given row and column in the matrix.

$$H : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \rightarrow R \qquad (\mathbf{1})$$

The initial terrain data can be a plain height-field or it can be imported from a file of real landscape data, e.g., a Digital Elevation Model *(DEM)* or a Geographic Information System *(GIS)*. It can be also modelled by any automated method, e.g., Midpoint Displacement, Poisson Faulting or Particle Systems-based (FOURNIER et al. (1982), MANDELBROT (1982), MUSGRAVE et al. (1989), MUSGRAVE (1993), SHANKEL (2000), WARSZAWSKI (2009), WARSZAWSKI and NIKIEL (2009)).

The hardness-field layer

The second layer *(D)* stores data describing the hardness of materials from which a given landscape was constructed and is a function defined in equation 2.

$$D : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \rightarrow R \quad (\mathbf{2})$$

This data can be obtained from geological records. Alternatively, for the virtual environment we propose to simulate it by modified *Poisson Faulting* (also known as *Fault Formation*) algorithm (FOURNIER et al. (1982), MANDELBROT (1982), MUSGRAVE (1993)).

HARDNESS SYNTHESIS

As in classic approach, we start generating hardness-field layer from 'flat ground' stored in a two-dimensional array. In the first phase we must select a value for height of the fault and a total number of them to be generated over the layer. Less number of faults bring more 'crispy' material clusters. To generate a fault, it is necessity to pick two different points with coordinates *(x_1, y_1) and (x_2, y_2)* exist in the hardness layer. Whit this points we can describe the line of the fault. Next, for each cell coordinates *(i, j)* in the hardness layer, we calculate distance from that line of fault (see equation 4) as determinant of the distance matrix *(S)* constructed as follow in equation 3:

$$S_{ij} = \begin{bmatrix} x_2 - x_1 & i - x_1 \\ y_2 - y_1 & j - y_1 \end{bmatrix} \quad (\mathbf{3})$$

that enables:

$$\det(S_{ij}) = (x_2 - x_1) * (j - y_1) - (y_2 - y_1) * (i - x_1) \quad (\mathbf{4})$$

The distance factor determines on which side of the given fault, the cell of the hardness layer is located (see figure 1).

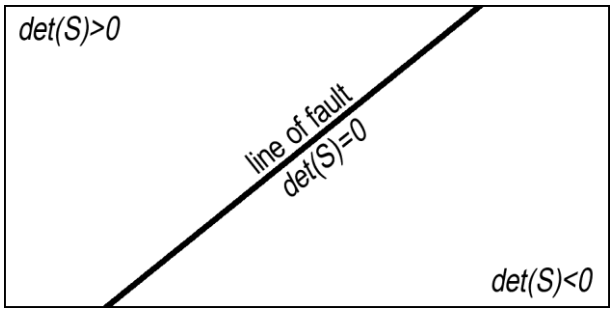

Fig. 1. Distance factor

After that step, the layer data is submitted to partial modifications with regard to distance factor *(det(S))*. If this factor is positive then value in given cell *(d)* increases with the previously selected height value of the single fault *(Δh)*. Otherwise, the cell value remains unchanged (see equation 5). The fault generation procedure is repeated until all faults are applied to the hardness-field layer.

$$d_{ij} = \begin{cases} d_{ij} + \Delta h & , \text{if } det(S_{ij}) > 0 \\ d_{ij} & , \text{if } det(S_{ij}) \leqslant 0 \end{cases} \quad (\mathbf{5})$$

The fault forming process results in a fractional Brownian surface *(fBs)* that is usually a base for further modifications (MANDELBROT (1982)).

To achieve that is very important to normalize each data to the interval *[0, 1]*. For that purpose, we use the equation 6, a simple normalization method, where normalized values *(d')* are calculated by modifications of initial records *(d)* at given coordinates *(i, j)* by minimal *(min)* and maximal *(max)* values derived from the entire non-normalized data structure.

$$d'_{ij} = \frac{d_{ij} - min}{max - min} \quad (\mathbf{6})$$

The next phase is to choose the total number of materials of which given landscape model is constructed. We called this factor *the class of hardness of terrain model*. When normalized data is less than value *(1.0)* then hardness value *(d'')* at given coordinates *(i, j)* is the floor function for multiplications of normalized data *(d')* and class of hardness *(λ) divided by it*, with restrictions stated in equation 7, for initial data equal to *(1.0)* value. This limitation eliminates artefacts data that creates excessed hardness records.

$$d''_{ij} = \begin{cases} \frac{\lfloor d'_{ij} * \lambda \rfloor}{\lambda} & , \text{if } d'_{ij} < 1 \\ \frac{\lambda - 1}{\lambda} & , \text{if } d'_{ij} = 1 \end{cases} \quad (7)$$

Finally, we achieve clusterized hardness data on the second layer of the terrain model. The general structure of presented technique is shown in figure 2.

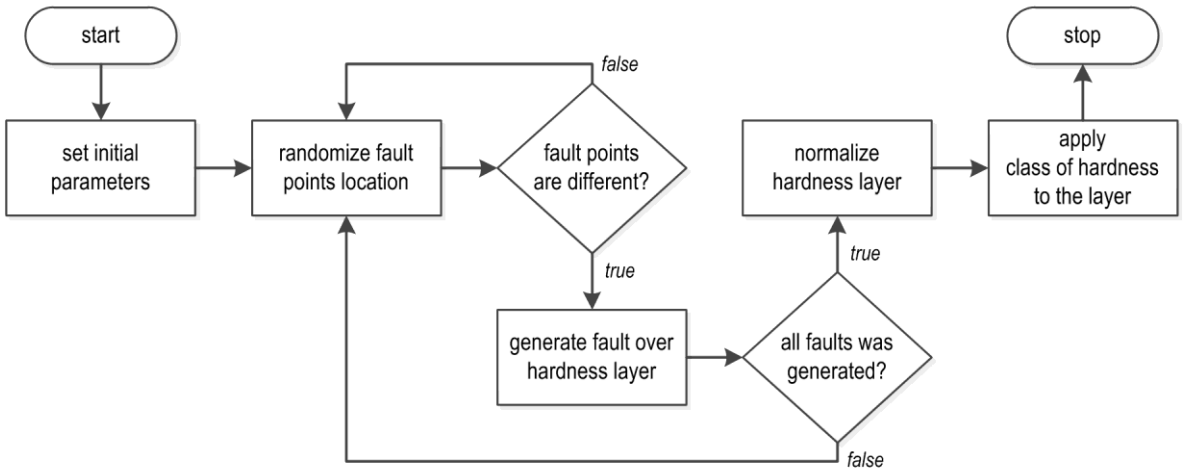

Fig. 2. Structure of the hardness synthesis algorithm

THE IMPLEMENTATION

The overall processing of the data structure is ideal for parallelization and implementation in CUDA environment. This 'parallelism' of the proposed technique results in near real-time synthesis of hardness model for height-field based landscapes. General CUDA implementations of main algorithm phases are shown in follow listings:

Listing 1. Poisson Faulting (phase one).

```
__global__ static void Faulting(float fSize,              // height of fault
                                float p1_X, float p1_Y,   // position of fault 1st point
                                float p2_X, float p2_Y,   // position of fault 2nd point
                                int cols, int rows,       // resolution of layer
                                float* layer)             // records of layer
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;    // cell index

    if (index < cols * rows)                              // index cannot exceed layer size
    {
        float p0_X = index % cols;                        // cell coordinates
        float p0_Y = index / cols;                        // in 2D space

        float det_S = (p0_X - p1_X) * (p0_Y - p2_Y)       // distance factor
                    - (p0_X - p2_X) * (p0_Y - p1_Y);      // and it calculation

        if (det_S) > 0)                                   // for each cell with positive data
        {
            *(layer + index) += fSize;                    // increase value of cell
        }
    }
}
```

Listing 2. Normalization (phase two).

```
__global__ static void Normalize(float min, float max,     // extreme values of layer
                                 int cols, int rows,       // resolution of layer
                                 float* layer)             // records of layer
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;     // cell index

    if (index < rows * cols)                               // index cannot exceed layer size
    {
        *(layer + index) = (*(layer + index) - min)        // modify cell data
                         / (max - min);                    // to normalized value
    }
}
```

Listing 3. Applying class of hardness (phase three).

```
__global__ static void Hardenize(int hClass,               // class of hardness
                                 int cols, int rows,       // resolution of layer
                                 float* layer)             // records of layer
{
    int index = blockIdx.x * blockDim.x + threadIdx.x;     // cell index

    if (index < rows * cols)                               // index cannot exceed layer size
    {
        if (*(layer + index) < 1.0f)                       // for each cell with value less than 1.0
        {
            *(layer + index) =                             // apply
                   ((int) (*(layer + index) * hClass))     // this modification
                   / (float) hClass;                       // to cell value
        }
        else                                               // in other case
        {
            *(layer + index) = (hClass - 1.0f)             // apply
                             / hClass;                     // this modification
        }                                                  // to cell value
    }
}
```

RESULTS

The resulting data structure can be simply assigned to the hardness values collected from Friedrich Mohs scale of mineral hardness. It can be used to simulate the erosion processes and their results reflected by virtual landscapes.

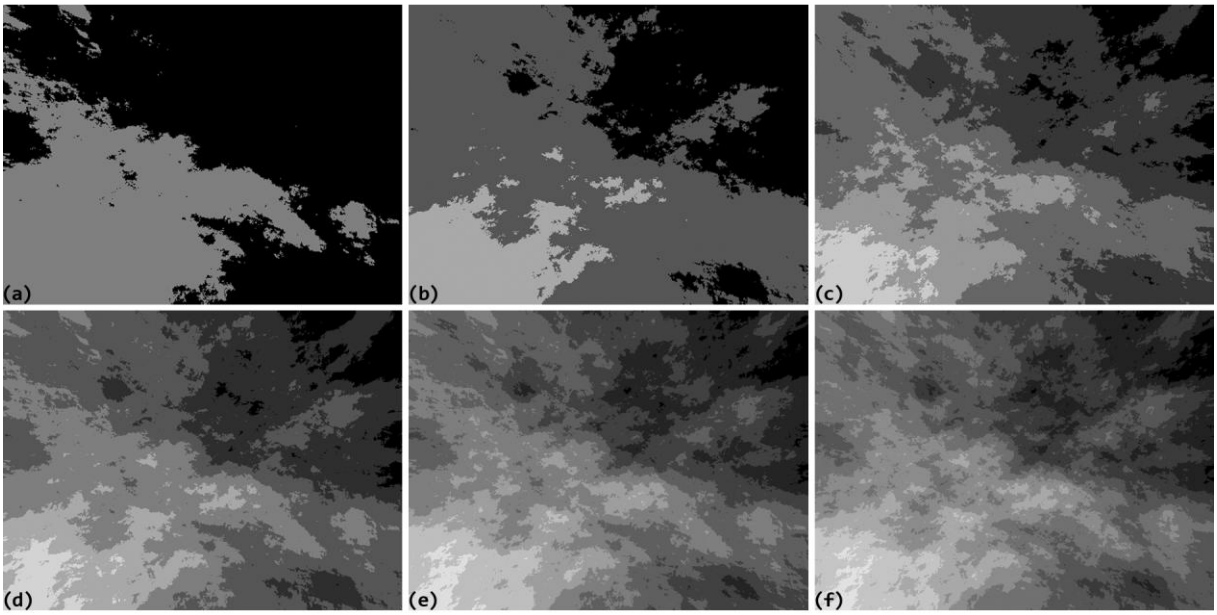

Fig. 3. Classes of hardness – graphical representation
(a) 2nd class model, (b) 3rd class model, (c) 5th class model,
(d) 6th class model, (e) 8th class model, (f) 9th class model

As is presented in figure 3, different materials are drawn in different tones of greyscale. The results of our simulations depicts formations of mineral structures similar to horizontal cross-section of a land. In direct relation to the class of hardness layer, we achieve more or less varied geological materials

The numerical complexity of the algorithm depends mostly on the number of used faults and decreases when this number increasing. The experimental results, as is shown in figure 4, are similar to the results received with classic Poisson faulting technique and confirm the linear nature of our method. The both normalization and mineralization phases have minimal influence (bellow 0,01 sec) to the overall performance of the hardness synthesis process and may be omitted in the performance calculations.

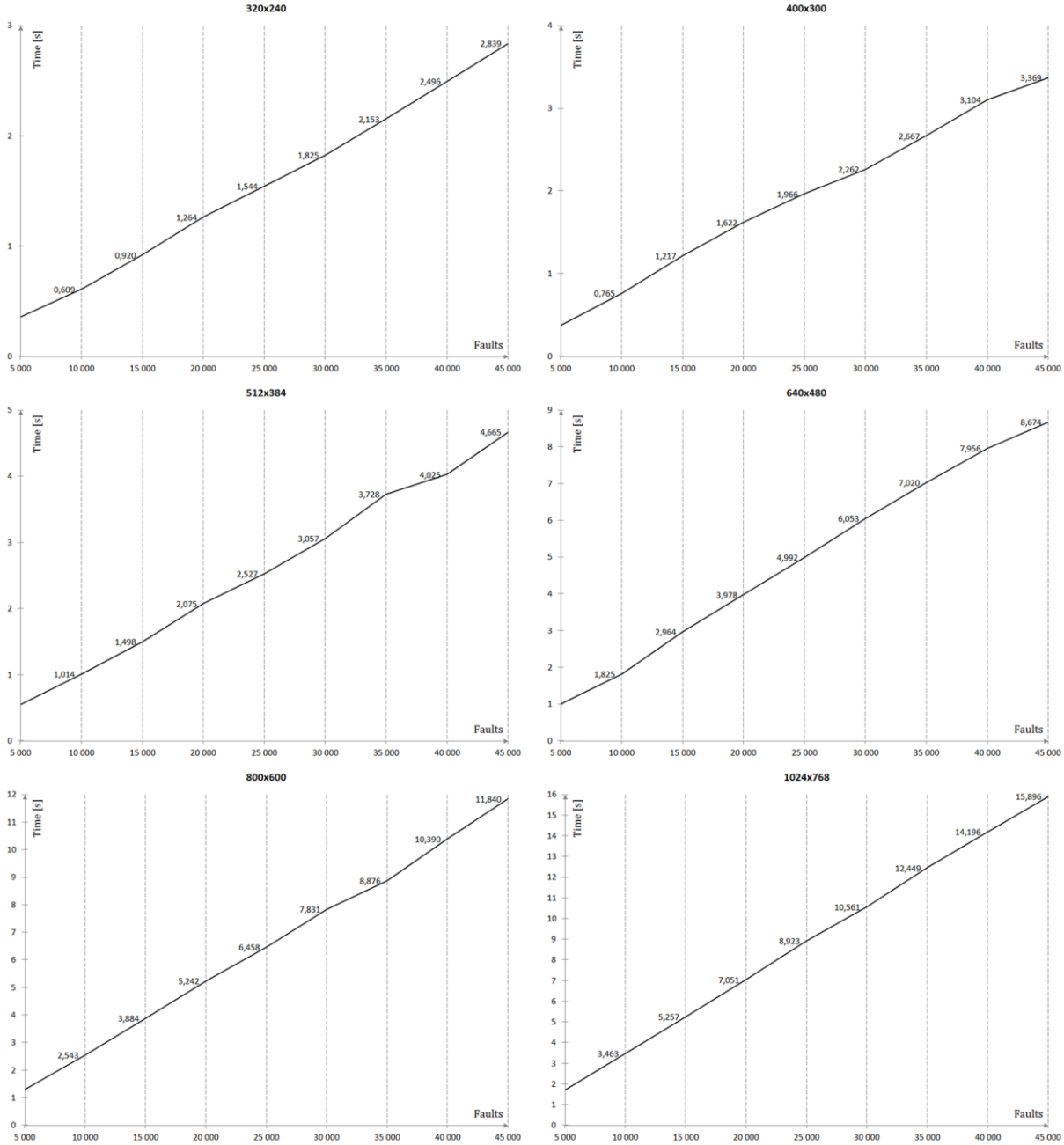

Fig. 4. Performance diagrams of hardness synthesis technique

All simulations processes were performed on CUDA-supported nVidia GeForce GTX 560 Ti class GPU. For selected resolution of hardness-field layer, the experimental procedure includes 1 000 generation of faults between 5 000 and 45 000. The diagrams correspond to the average values obtained in those simulations.

CONCLUSION

In the paper, we proposed the technique for automated hardness synthesis for height-field based terrain models. Such a model opens efficient way for implementing mineral base in the general landscape structure. The performance of the algorithm shows the possibility to make numerous generations of different internal geological structures for a single terrain model. It can serve as a basis for further simulations of erosion forces acting on that terrain with regard to different mineral depositions. Wide class of hardness data can be done in near-real time process. The use of Poisson Faulting algorithm as a mathematical background of our proposition helps to generate similar hardness data in areas close to each other. The main advantage is clusterized data instead of randomly displaced 'noisy' depositions, thus it is similar to natural structure which can be observed in cross-sections of a land. Additionally, the hardness-field can be directly adopted to terrain models based on the LDR data structure.

Our further research will be focused on the development of hardness synthesis method for landscapes based on the voxel representation, that can be used for improvement of erosion simulation on fully three-dimensional terrain models.

REFERENCES

BENES, Bedrich and FORSBACH, Rafael (2001) Layered data representation for visual simulation of terrain erosion. Proceedings of 17th Spring Conference on Computer Graphics (SCCG '01), Budmerice, Slovakia, 25-28 April, IEEE Computer Society, pp. 80-85.

BENES, Bedrich and FORSBACH, Rafael (2002) Visual simulation of hydraulic erosion. Proceedings of 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '02), Plzen-Bory, Czech Republic, 4-8 February, pp. 79-86.

BENES, Bedrich (2007) Real-time erosion using shallow water simulation. Proceedings of 4th Workshop in Virtual Reality Interactions and Physical Simulation (VRIPHYS '07), Dublin, Ireland, 9 November, pp. 43-50.

BOHEMIA INTERACTIVE SIMULATIONS (2011), http://www.bisimulations.com

BONK, Curtis J. and DENNEN, Vanessa P. (2005) Massive multiplayer online gaming: A research framework for military training and education, TR 2005-1, Office of the Under Secretary of Defense for Personnel and Readiness, Washington, USA.

CLEMPNER Julio B. and POZNYAK Alexander S. (2011) Convergence method, properties and computational complexity for Lyapunov games. International Journal of Applied Mathematics and Computer Science, 2 (21), pp. 349-361.

DI TRAPANI Lyall Jonathan and INANC Tamer (2010) NTGsim: A graphical user interface and a 3D simulator for nonlinear trajectory generation methodology. International Journal of Applied Mathematics and Computer Science, 2 (20), pp. 305-316.

FOURNIER, Alain, FUSSELL, Don and CARPENTER, Loren (1982) Computer rendering of stochastic models. Communications of the ACM, 6 (25), pp. 371-384.

FRADE, Miguel, DE VEGA, F. Fernandez and COTTA, Carlos (2009) Breeding terrains with genetic terrain programming: The evolution of terrain generators. International Journal of Computer Games Technology, 2009, pp. 1-13.

LENGYEL, Eric Stephen (2010) Voxel-based terrain for real-time virtual simulations (PhD thesis). University of California, Davies, USA.

MANDELBROT, Benoit B. (1982) The fractal geometry of nature (2nd ed.). H. W. Freeman and Co., San Francisco, USA.

MUSGRAVE, F. Kenton, KOLB, Craig E. and MACE, Robert S. (1989), The synthesis and rendering of eroded fractal terrains. Proceedings of 16th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '89), Boston, USA, 31 July - 4 August, ACM, pp. 41-50.

MUSGRAVE, F. Kenton (1993) Methods for realistic landscape imaging (PhD thesis). Yale University, New Haven, USA.

RICKEL, Jeff and JOHNSON, W. Lewis (1999) Virtual humans for team training in Virtual Reality. Proceedings of 9th World Conference on AI in Education (AIED '99), Le Mans, France, 19-23 July, IOS Press, pp. 578-585.

SHANKEL, Jason (2000) Fractal terrain generation - Fault formation. In: DeLoura, Mark A. (eds), Game Programming Gems (vol. 1). Charles River Media, Hingham, USA, pp. 499-502.

SHANKEL, Jason (2000) Fractal terrain generation – Midpoint displacement. In: DeLoura, Mark A. (eds), Game Programming Gems (vol. 1). Charles River Media, Hingham, USA, pp. 503-507.

SHANKEL, Jason (2000) Fractal terrain generation – Particle deposition. In: DeLoura, Mark A. (eds), Game Programming Gems (vol. 1). Charles River Media, Hingham, USA, pp. 508-511.

SMELIK, Ruben, TUTENEL, Tim, DE KRAKER, Klass Jan and BIDARRA, Rafael (2010) Declarative terrain modeling for military training games. International Journal of Computer Games Technology, 2010, pp. 2:1-11.

WAN, Ming, QU, Huamin and KAUFMAN, Arie (1999) Virtual flythrough over a voxel-based terrain. Proceedings of the 1st IEEE Virtual Reality Conference (VR '99), Huston, USA, 13-17 March, pp. 53-60.

WARSZAWSKI, Korneliusz (2009) Ground from smoke: Using particle systems in C#. Game Developer Magazine, 3 (16), pp. 15-21.

WARSZAWSKI, Korneliusz and NIKIEL, Slawomir (2009) A proposition of particle system-based technique for automated terrain surface modeling. Proceedings of 5th International North American Conference on Intelligent Games and Simulation (Game-On-NA '09), Atlanta, USA, 26-28 August, EUROSIS, pp. 17-19.

WELLS, William D. and DARKEN, Christian J. (2005) Generating enhanced natural environments and terrain for interactive combat simulations (GENETICS). Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '05), Monterey, USA, 7-9 November, ACM Press, pp. 184-191.