# Layered Data Representation for Visual Simulation of Terrain Erosion

Bedřich Beneš *  Rafael Forsbach
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México

## Abstract

*New data structure for visual simulation of 3D terrains is introduced. The representation is inspired by real geological measurements and presents good trade-off between commonly used inexpensive, but inaccurate, height fields and memory demanding voxel representation. The representation is based on horizontal stratified layers consisting of one material. The layers are captured in certain positions of the landscape. This representation is then discretized into 2D array.*

*We demonstrate that the classical algorithm simulating thermal erosion [10] can run on this representation and we can even simulate some new properties. The simulation has been done on artificial data as well as on real data from Mars.*

**Keywords:** Terrain erosion, layers, height field, and voxels.

## 1 Introduction and Previous Work

One of the disadvantages of the fractal surfaces is their inability to capture terrain evolution – erosion. This problem has been well known since 1982 [7] and many approaches have been devoted to diminish or eliminate this their property [1, 2, 3, 4, 6, 8, 10, 11, 12]. The principles of these techniques are ranging from pure *ad hoc* approaches to physical or at least physically based simulations. The techniques based on physics are getting closer to the simulation of reality and the algorithms can be therefore used in geology and related areas.

Probably the first algorithms for visual simulation of erosion were introduced in [10]. Musgrave has introduced two algorithms here – thermal and fluvial erosion and used these simulations to erode terrains generated by fractal techniques. This paper deals with the thermal weathering introduced here so this technique is described in depth in Section 3.

* beda@campus.ccm.itesm.mx

In the same publication Kelley et al. [6] introduced an *ad hoc* algorithm that goes from the opposite way. Instead of eroding fractal terrain, they generate landscape that corresponds to underlying structure of water streams. Fractal interpolation is extensively used here to generate the hills connecting the water streams. The work [8] is trying to formalize some of the algorithms used for simulation of erosion under one algorithm that is based on rewriting the matrices. The authors define classes of matrices that are used for different erosion algorithms and demonstrate their ability to simulate existing techniques.

The latest works of Musgrave [5, 10, 11] are not primarily focusing erosion-based models but describe techniques used for modeling landscapes by blending some well-defined and elaborated noise functions (Perlin, fBm, etc.) These approaches give visually plausible results, but there are not known any facts that relate these techniques to reality. The author is claiming that the erosion is much stronger tool, but the number of parameters and the time that is necessary to run the simulation makes these techniques practically unusable. The time demands of these techniques have been focused in [1]. The authors are using a semi-adaptive algorithm that is leaving the non-important parts and is eroding only the areas with high gradient or importance. Another class of algorithms deals with running water. The second algorithm of Musgrave et al. [10] supposes that the running water is dissolving some material and moving it to different location. This material is then deposited. Material at every point has certain parameters classifying these abilities. The water flows in the direction of the highest gradient as in the previous case. Another physically based model has been introduced by Chiba et al. [2]. The speed of the running water defines so-called velocity fields. These fields, together with the volume of water, are responsible for the forces that are then used to deposit certain amount of the material from place to place.

The voxels have been used for simulation of weathered stones in [4]. A similar approach, based on morphological operators in voxel space, has been recently published [12].

The techniques described above usually work with two kinds of data structures - height fields or voxels. The voxel-based techniques have highest precision and they are giving the best results. On the contrary, the height fields are simulating only surface erosion but algorithms running on this representation are usually faster. The sets of triangles [3] are usually used for fast rendering. The relation between the speed of the algorithms and the quality of representation is well known. The better is slower but captures more details and is more precise. We are proposing another technique here. Geological core samples and the ground structure have inspired the approach [9]. In reality the ground is structured in stratified layers. This fact is leaving the voxel-based representation that allows very abrupt changes of the structure, as too strong, but also too slow, tools. The height fields are not keeping representation of the underlying structures and are supposing the entire terrain consists of one kind of the material. We propose compromise representation here. This data structure is consisting of vertical intervals of equal density of material. This is keeping the advantage of voxel representation but eliminates the worst disadvantage - high computational time. We demonstrate the use of this data representation for simulating the thermal erosion [5, 10] in this paper. Any of the other erosion techniques [1, 2, 10] can be used with this data structure as well. This paper has the following structure. Section 2 describes classical data structures used for representation of the artificial terrains. The next section deals with the erosion model itself and Section 4 introduces new terrain representation. Thermal erosion used on the new data representation is explained in Section 5 and the next section describes some problems with visualization. The last two sections deal with some implementation issues, results, and conclusions.

## 2 Classical Terrain Representations

The focus of computer graphics approaches is in the visual plausibility. Many techniques are based on interactivity [5], but if we want to run real simulation, we should use real data and run algorithms that are simulating nature. Two kinds of representations are typically used for these purposes in computer graphics- height fields and voxel data. The first mentioned could be captured as two-dimensional matrix. Every element carries information about the height (this gives the name height fields), but also many other parameters; like amount of water kept there, ability of the

material to be dissolved in water, amount of minerals, soils, etc. In these data structures we suppose that the entire column is consisting of one kind of material. This is the most important simplification. The voxel based representations [4, 12] are dividing terrain or objects into 3D cubes - voxels. The information kept in every voxel is similar to the one kept in one element of the height field - material, amount of water, resistance in the environment, etc. The space requirements are apparent. Suppose $n$ bytes of data are kept for each element. For a height field of 1024x1024 elements we need $n$ MB of data. For the voxel array in similar resolution we need $n$ $1024^3$ i.e., $n$ GB. From this viewpoint the height fields are better. On the other hand the algorithms for erosion simulation with height fields do not provide some important things like meanders or digging horizontal caves. This everything can be done in voxels.

## 3 Thermal Weathering

Erosion algorithms can be described by the material transport among terrain elements. This is captured by the following differential equation:

$$Q_{in} - Qout = \frac{dS}{dt}$$

where $Q_{in}$ and $Q_{out}$ is the input and the outlet of the element respectively. $S$ is the volume inside the element and $t$ is time. Different algorithms use different approaches to solve this equation. The flow of the material depends on the phenomena that the algorithms simulate.

In this paper we use thermal weathering originally introduced in [10]. This algorithm is dealing with long-term thermal erosion. The material is dissolved because of changes in temperature. This is typical for example the case of Moon's craters or in the areas of high amplitude of temperature. Due to the thermal shocks the small parts of the terrain are breaking-up and falling down. Depending on the consistence of the material, this process is faster or slower. The eroded part is falling down in the direction of the greatest gradient.

Let's denote the height of the investigated element by $h$ and its eight neighbors by $h_i$ $i = 1, 2, ..., 8$. We denote the height difference between the currently investigated element and the lowest neighbor by $H$. This is the maximum, so $H = maxh - h_i, i = 1, ..., 8$.

The area of the elements is a and the volume to be moved ($\Delta S$ is therefore $\Delta S = aH/2$. We cannot move more otherwise the algorithm will oscillate. This amount is moved to the neighbors proportionally. Let's denote the set of the neighbors that are lying lower than the central element by $A = h_i, h_i - h < 0, i = 1, ..., 8$. Then each element from the set A will get part of the volume ($\Delta S_i$ proportional to its height difference, i.e.,

$$\Delta S_i = \Delta S \frac{h_i}{\sum_{\forall h_k \in A} h_k}$$

At this point random numbers play important role. To simulate randomness we affect the total amount of the material to be moved. We decrease this amount by at most ten percent of the volume. The random number generator

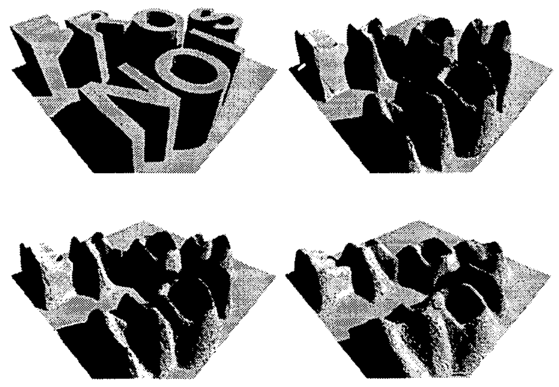

Figure 1. Example of thermal erosion

we use has Gaussian distribution that better corresponds to nature.

To simulate viscosity of the material, so called talus angle $\alpha$ is defined. If the given gradient is smaller than this limit, the material is not moved to this position. The distance between two elements is constant, let's denote it by $d$. Then the $\alpha = \tan h - h_j/d$. The condition of the talus angle reduces the set of elements that will obtain some portion of the volume to $A = \{h_j, h_j - h < 0 \wedge (h - h_j)/d > \tan^{-1}\alpha\}$. Figure 3 demonstrates output one simulation with the talus angle set to 45°. The important fact, from our point of view, is that these algorithms are dealing with the data structures described above. The new data representation perfectly fits with these algorithms.

## 4 New Terrain Representation

In geology the data obtained by measuring by core samples can be the best represented as a layered structures. Figure 4 schematically shows typical data obtained by this measurement. The sample of the layers is consisting of material that has certain density, amount of dissolved minerals, water, amount of gas, etc.

This data is usually obtained at different locations and then interpolated [6]. The interpolated data defines 3D field that can be sampled into voxel array or its surface can be used for a height field.

Results of these measurements and calculations are layers of different material as they are distributed under the ground and on the surface. Using voxels for representation of this data is a waste of data because the layers are usually very thick. On the other hand using just a surface for a height field means discarding a lot of important information. Considering these facts, the apparent data structure that can be used is some kind of interval data structure that can be

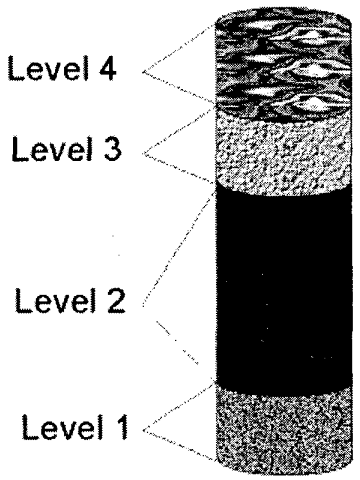

Figure 2. Typical structure obtained by geological core sample

thought as RLE in voxels. We are proposing the following.

We save the entire landscape as a two dimensional array. Every element of the array is consisting of one-dimensional array consisting of elements containing information about all underlying layers. So the landscape is 2D array of 1D arrays. The data structure used for one element of the landscape in our implementation is:

```
typedef struct{
      PropertiesT data[MAX_LEVEL];
      float height;
} ElmT;       //one element of the array
```

We limit the number of layers to MAX_LEVEL that is in our implementation ten. The properties within one layer are supposed to be constant. We keep here all the information described above (water, density, gas, ability to deposit material, saturation coefficients etc.) This data structure can be enriched arbitrarily. We can also use linked list of pointers to avoid limit of the maximal number of elements allowed. The height of the layer together with the index within the 2D array gives also information about the total height. Let's say, that the 2D array is defined as

```
ElmT terrain[X][Y];
```

Then the height $h$ in point

```
terrain[i][j]
```

is a sum of the heights of all the layers i.e.,

$$h = \sum_{k=0}^{MAX\_HEIGHT-1} terrain[i][j].data[k].height$$

The height of the column is important for the visualization and conversion to data structures that can be easily rendered (see Section 6). We suppose the levels that are not used have zero height. In this case the formula described above is right, but from the viewpoint of the speed of the algorithm this can slow down the speed. In our implementation we keep additional information about the last level used. It is important to note that the complexity of the algorithms is equal to the ones running on voxels i.e., $O(n^3)$. In reality instead of traversing $n^3$ elements we have to go through $k.n^2$, where $k$ is the number of the layers - this is usually much smaller and it depends on the thickness of the layers. Another important fact should be pointed out here. Some layers can have density set to zero, or include gas or water. In other words they can represent caves and holes. This is an advantage of this representation inherited from the voxels that cannot be captured by height fields.

## 5 Thermal Erosion Applied to the New Terrain Representation

We have implemented the thermal erosion algorithm [10], originally running on height fields, on the newly proposed data structure. The original algorithm has been proposed on height fields so we have also enriched this algorithm by some new features.

Implementation works in the way the original algorithm is described. We compute gradient – this is a more complicated in our case. We have to evaluate height of all layers in a given element and compare it with the neighbor. Then we evaluate the angle and if this is greater than the given talus angle the algorithm erodes this element. We compute the amount of material to be moved from the upper layer. If we need to move more material than the layer contains we have to elimiate the layer, i.e., set it to zero height, and continue with the layer that is under this one.

The important thing is that this erosion is not applied only to the surface of the terrain, but also in all subsurface holes. Another fact, the falling of material from the ceiling of the caves, is also captured here. To simulate this we have to find all layers with zero density and apply the algorithm to the layer that is under the zero density layer.

We have also implemented one new property that cannot be captured in the height fields. Usually the non-deposed material is very dense, because it is at the place for long time. When eroded, the material that moves is changed to dust that has completely different properties. We have easily captured this in our algorithm setting the material as easily movable i.e., changing its density. At the moment the material is dissolved from very hard object, it continues its depositing easily.

## 6 Visualization

Actually, we use free-ware Persistence of Vision raytracer (POV) for displaying the data. The POV works with height-fields, or meshes of triangles, so we have to save all data as these representations. The images generated from height fields have certain signature and cannot display caves and underground structures even in cuts.

Better visualization would involve techniques like marching cubes algorithm adapted to this data structures (see for example [13]) to transform our data into set of triangles that can be rendered efficiently. Possibly we can also use any algorithm that will first find all the empty spaces and take only their surfaces.

## 7 Implementation and Results

We have implemented the algorithm in C and we run it on PC equipped with PentiumIII/500MHz. To better explain the next part, by one erosion step we assume rewriting of the data structure using thermal erosion, i.e., decision in every point how much of the material will be moved and doing this. One hundred erosion steps of a terrain in resolution 1024x1024 elements and five layers runs 239 minutes. Figure 7 shows one artificial example. Letter W consisting of very hard material is covered by mud that can be eroded easily. As the algorithm runs this material is eroded out and the shape of the letter W remains unchanged. This simulation cannot be done using height fields. The resolution of the underlying array was 400x400 elements in five layers and the algorithm run 1000 steps approximately 20 minutes.

Another example in Figure 7 shows real data of the volcano Olympus Mons. We have obtained the data from NASA - MOLA (Mars Orbiter Laser Altimeter) project. We have covered the top of the mountain by ice and we are simulating the water floating down from the dissolved solar cap. The mountain itself has bright color and the water is displayed as darker. This simulation has been done in resolution 750x750 elements in five layers. The algorithm run 500 erosion steps a little more than one hour. Figures 8 and 8 are the side and the top view of the similar simulation. Animations from these simulations can be found at http://paginas.ccm.itesm.mx/ beda/research/sccg01.html.

## 8 Conclusions and Future Work

New data structure for visual simulation of synthetical terrains has been introduced. The technique is inspired by real geological measurements and is good trade off between inexpensive but inaccurate height fields and memory demanding voxel representation. The representation is based on layers of equal materials. Moreover we have shown that classical algorithm for thermal erosion can run on this data. This algorithm can even capture some more realistic

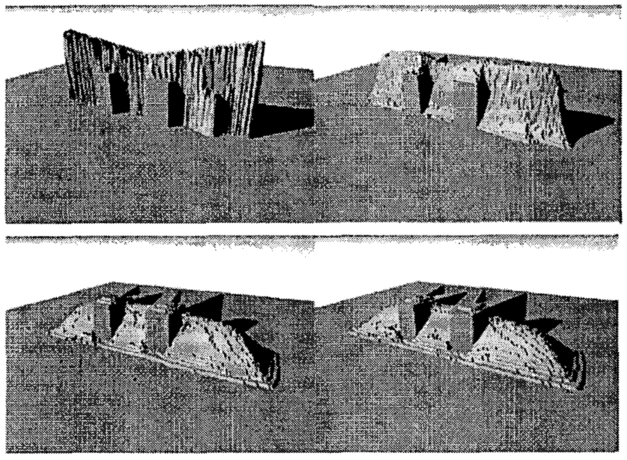

**Figure 3. Erosion of the hardly erodable letter W covered by a weak material that is easily eroded away**

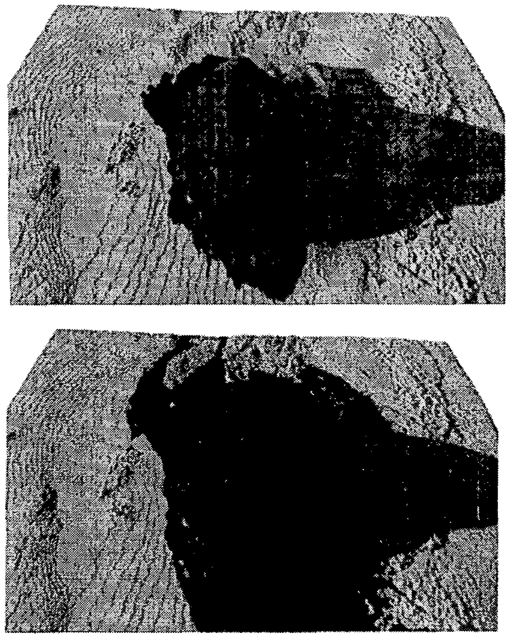

**Figure 4. A simulation of Olympus Mons. Water dissolved from the cap forms distinct rivers (upper part and side part of the image) in the canyons where it probably flown**

features, like subsurface erosion in caves and erosion of material consisting of different densities.

As a future work we would like to implement the algorithm in parallel. The simple divide-and-conquer technique, where every CPU is computing just a part of a data and at the end the material transferred through the boundaries is recomputed, is going to be implemented. The algorithm itself as well as the data is excellent for this task. Here every computational unit can keep just part of a landscape and two units are sharing just a boundary. Moreover the time needed for every erosion step is constant. Another challenging task is better displaying of the data. The graphics cards of computers and the standards for real-time rendering like OpenGL are "triangle native". We would like to implement some conversion of these interval data to set of triangles and to allow interactive manipulation with them. Another open problem is slumping of the landscape because of inner caves and gaps. Water going under the surface is causing many changes in the structure of the landscape and this can be probably easily captured as well. Last but not least, we would like to implement some other erosion techniques. First we want to implement existing techniques [2, 4, 9, 12] (modified to benefit from the new representation) and then we want to capture some new effects like motion of big, heavy, and firm objects (stones, rocks) on sliding and unstable underlying layers, etc.

**Acknowledgments:**

This work has been supponted by CONACyT and by grant ITESM.

## References

[1] B. Beneš, I. Marák, P. Šimek and P. Slavik. Hierarchical Erosion of Synthetical Terrains. *Proceedings of Spring Conference on Computer Graphics - SCCG'97*, Comenius University Bratislava, pp. 93-100.

[2] N. Chiba, K. Muraoka and K. Fujita. An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain Scenery. *Journal of Visualization and Computer Animation* 9, 1998, pp.185-194.

[3] A.R. Dixon and G.H. Kirby. Data Structures for Artificial Terrain Generation. *Computer Graphics Forum* 13, 1994, pp.73-48

[4] J. Dorsey, A. Edelman, H.W. Jansen, J. Legakis, and H.K.Pedersen. Modeling and Rendering of Weathered Stone. *Proceedings of SIGGRAPH 1999*, pp.225-234

[5] D.S. Eckbert, F.K. Musgrave, P. Prusinkiewicz, J. Stam and J.Tessendors Simulating Nature: From Theory to Applications. *SIGGRAPH 2000 Course Notes, 2000*

[6] A.D. Kelley, M.C. Malin and G.M. Nielson. Terrain Simulation Using a Model of Stream Erosion. *Proceedings of SIGGRAPH 1989*, pp.263-268

[7] B.B. Mandelbrot The Fractal Geometry of Nature. W.H.Freeman, 1982

[8] I. Marák, B. Beneš, and P. Slavik. Terrain Erosion Based on Rewriting of Matrices. *Proceedings of The Fifth International Conference in Central Europe on Computer Graphics and Visualization - WSCG*, 1997, pp.341-351

[9] D. Merritts, A. de Wet, and K. Menking Environmental Geology. W.H. Freeman and Company, 1998

[10] F.K. Musgrave The Synthesis and Rendering of Eroded Fractals Terrains *Proceedings of SIGGRAPH 1989*, pp.11.1-11.9

[11] F.K. Musgrave Towards the Synthetic Universe. *IEEE Computer Graphics and Applications*, November-December 1999, pp.10-13

[12] N. Ozawa, and I. Fujishiro A Morphological Approach to Volume Synthesis of Weathered Stones. *Volume Graphics*, M.Chen, and A.Kaufman, and R.Yagel (editors), Springer-Verlag, 2000, pp. 367-378

[13] A. Watt and M. Watt Advanced Animation and Rendering Techniques: Theory and Practice Addison-Wesley Readings, 1992

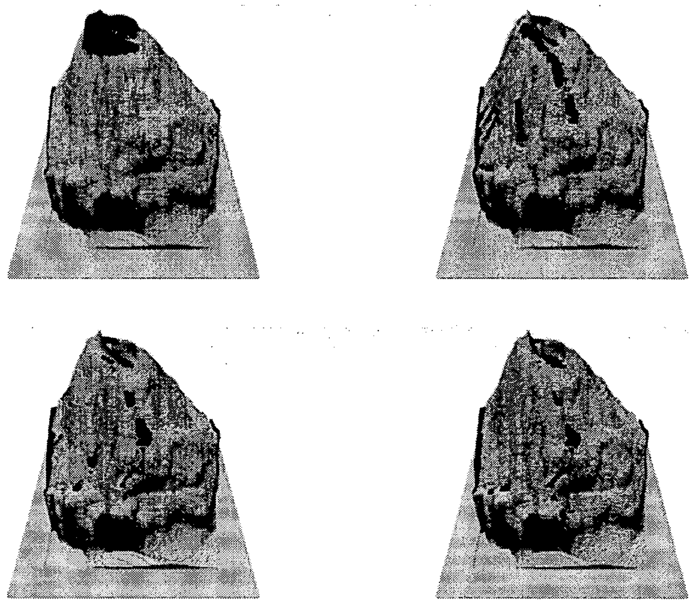

Figure 5. Simulation of erosion of Olympus Mons. The polar cap is dissolved and the water running down is displayed darker than the mountain itself

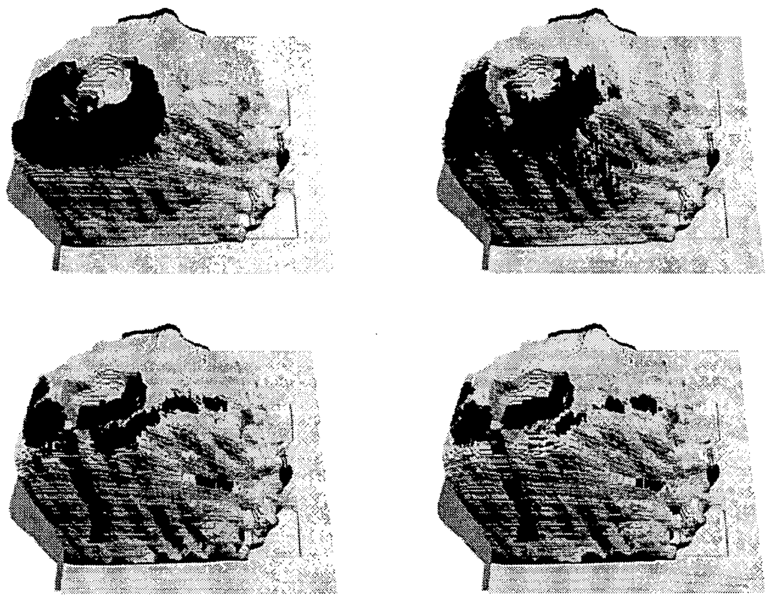

Figure 6. Top view of the simulation from the previous image. The crater of the volcano is filled by water