# Visual Simulation of Hydraulic Erosion

**Bedřich Beneš**
**Rafael Forsbach**

Department of Computer Science
ITESM, Campus Ciudad de México
Mexico D.F.
beda@campus.ccm.itesm.mx

**ABSTRACT**

A new algorithm for hydraulic terrain erosion is introduced. The main goal of the paper is to provide a technique which is inspired by physics and which allows for high level of control. We divided the erosion process into four independent steps that can be applied independently to achieve high level of realism. The erosion algorithm is based on the ability of water to dissolve material that is then transported to another locations. Because of the evaporation the sediment capacity of water volume is exceeded and the material is deposited. This kind of material transport significantly influences the terrain morphology. The algorithm has wide area of applications; we describe here water sources, drying up the plashes, as well as fictive rain on the surface of Mars.

**Keywords:** physics-based modeling, erosion processes, weathering, visual simulation

## 1 Introduction

Terrain erosion techniques have been addressed by computer graphics scientists for more than twelve years. The algorithms introduced and techniques so far are mostly *ad hoc*, or more or less inspired by hydrodynamics and sediment transport models borrowed from physics.

Musgrave *et al* [11] have described one of the first algorithms visually simulating terrain erosion. Two algorithms have been introduced in this paper – thermal weathering and hydraulic erosion. The first algorithm simulates the sediment runoff caused by thermal shocks of the terrain. Part of the material is simply deposited on different locations, depending on the local gradient of the surface. The later technique is based on the fact that water can dissolve, transport, and deposit certain amount of soil. Depending on the gradient of the terrain location, some amount of material is dissolved and moved in water. This material is later deposited in another location.

Chiba *et al.* [3] have introduced another physics-based algorithm. This simulates ridges and valleys by applying forces caused by the running water to the terrain surface. Water is approximated by particles and a simple collision detection algorithm is used to simulate the erosion.

A similar technique has been described by Nagashima [12]. The main difference is that the terrain is eroded adaptively. The shape of the river is generated independently to the surface using two-dimensional fractal interpolation. Then the banks of the river are eroded using physics-inspired rules.

Simulations of weathered stones has been described in [5, 15]. These new techniques are mostly based on the following idea: first, the three-dimensional object is covered by a layer of voxels. These voxels are the subject of material transport that is the solution of differential equations. The equations usually model some kind of diffusion or soil transport in the material. Two important things are usually simulated, surface fall-off and sediments coming out from the object. High effort is also devoted to the rendering of these objects.

The height field is the most commonly used data structure for the terrain simulation and visualization [3, 10, 11, 12]. The advantage of this data representation is that it can be easily ray-traced and does not need a large storage space. On the other hand the voxel representation [5, 15] is more precise, allows simulation of underground structures such as caves, but requires much more memory space. Beneš *et al.* [1] have introduced a layered data structure that fits with the problem of terrain erosion and presents a good trade-off between height fields and voxels. The set of triangles is another common representation used in terrain modeling [4, 9]. This data structure is convenient for fast displaying (see [6, pp:369–404]) and is also frequently used in GIS.

The previously described erosion algorithms focus on the erosion process itself but not the underlying mechanisms that allow it (water evaporation, water sources, material properties, etc.). The algorithms are usually quite complex, the techniques are implicit, and not well related to physics. They also use large numbers of constants that influence each other and a stable implementation of these techniques can be a hard problem. Even worse, some algorithms tend to oscillate, due to the simple models of water transport.

The main goal of this paper is to provide a fast, stable, easy to use, and easy to control technique for visual hydraulic erosion simulation. We use a technique that is based on the physical models of [2] and [14]. However, the main goal of the paper is visual plausibility and not the physical correctness. The technique presented here is sufficiently fast to be implemented in any interactive modeler.

We have separated the hydraulic erosion process into four independent steps that can be applied repeatedly and in arbitrary order to achieve desired level of realism. These steps are described with certain precision and better algorithms and techniques can be the possible extensions.

The paper is structured as follows. In the next section we describe the process of hydraulic erosion and factors that influence it. Section 3 describes implementation and results of our simulations. The last Section 4 describes some opened questions and conclusions of the paper.

## 2 Hydraulic Erosion

The process of hydraulic erosion consists of several distinct steps. First, water appears at some place or places. This can be due to the presence of rain, water sources or because of the water flow. Water absorbs the material, either because of the flow that erodes the surface or because of the dissolving. Next, water, as well as the captured sediment, is transported according to inner and outer forces. The most important factor contributing to the water flow is gravitation, although the internal forces also play an important role. The water drops the carried and the suspended material at certain locations. This deposition process is influenced by two mayor factors [14]. First, the water slows down so the heavy particles of material cannot be carried anymore. The second factor contributing to the deposition process is the excess of the water sediment capacity. This is caused by the evaporation of the water.

In the previous work these processes were simulated as related, for example in [11] the material deposition is a function of the water transportation, but they can also be treated independently. The hydraulic erosion process can therefore be described by the following fours independent steps:

1. new water appears,
2. water erodes the underlying terrain and captures the material,
3. water and the suspended material are transported, and
4. water deposit the material at another locations.

We use the layered data structured described in [1] in our implementation, but here, for simplicity, we will work with height fields as we describe the algorithm

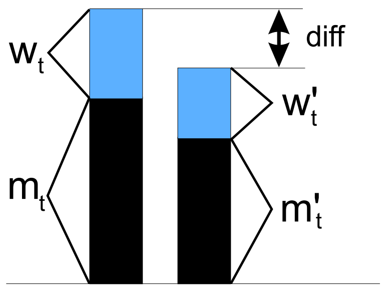

Figure 1: Notation of the vertices and the material. The left column is the explored vertex whereas the right one is a representative of the neighboring vertices.

Let $m_t$ be the height of the material (see Figure 1) at a given position (vertex of the height field), let $w_t$ denote the volume of water and let $s_t$ be the amount of dissolved soil at the given time $t$. The next state of these variables is denoted $m_{t+1}$, $w_{t+1}$, and $s_{t+1}$. The coefficient $t+1$ does not explicitly mean time increased by one second, it can be thought as $t+\Delta t$, so the increment means an arbitrary time step, or simply another state of the system.

The system distributes water and soil to the neighbors. For the sake of clarity let us suppose just one neighboring vertex and let us denoted the corresponding quantities in by $m'$, $w'$, and $s'$. The problem of the full two-dimensional distributions is discussed in the Section 2.4.

## 2.1 Rain and Water Sources

Water can either appear locally at certain position (water sources) or it can affect large areas of the terrain in the form of rain. In both cases the level of water in the element is simply increased by $K_r$ that is the amount of water that has arrived in the time step $[t, t+\Delta t]$. This is described by the following equation:

$$w_{t+1} = w_t + K_r.$$

The coefficient $K_r$ can be constant, corresponding to the constant inflow of water. To achieve better simulation, this coefficient should respect time and the position $\mathbf{P}$ in the space $K_r(t, \mathbf{P})$. In nature, the rain intensity at a given point over time corresponds to a bell shaped function [14] and the locality can be described by a fractal functions [11].

## 2.2 Evaporation

The amount of evaporated water depends on the temperature and the extent of the water level. Since we use height field representation, the area is constant for each element. We also suppose the temperature to be constant (corresponding to the average over a large time interval). The evaporation of the water is described as:

$$\frac{dw}{dt} = -K_e w, \qquad (1)$$

where $K_e$ is the evaporation coefficient, corresponding to the speed of the water evaporation in the given time span. Solution of the equation (1) shows that the amount of water presented at a given vertex depends only at the initial water volume $w_0$ and time. The amount of water decreases exponentially.

$$w_t = w_0 e^{-K_e t}. \tag{2}$$

In practical implementation the level of water will never reach zero, se we modify this function by introducing a threshold value $T$. If the amount of water decreases under this level, we set it to zero. This allows us to simulate drying up of some areas. The modified equation (2) has form:

$$w_t = w_0 e^{-K_e t} - T.$$

### 2.3 Deposition

We do not consider erosion based on forces in this paper, so in the following we will discuss only deposition of the sediment and soil that is dissolved in water.

Deposition of the material captured in slowly moving water has two causes in the nature. First, the material deposits because the moving particles of the material reach the ground and cannot be captured anymore. We do not simulate this here. Description of this kind of deposition used in computer graphics can be found in the paper [11].

We use another technique. The water volume has certain amount of the dissolved soil that cannot exceed the saturation level. The saturation level $S_t$ is described by the equation

$$S_t = K_s w_t, \tag{3}$$

where the $K_s$ is the saturation coefficient in kilograms per liters. The coefficient says that one liter of water can carry $K_s$ kilograms of material. If some volume of water evaporates, the saturation level will be exceeded so the corresponding amount of the material must be deposited at the given position. We evaluate the amount of material that can be dissolved using the equation (3) and compare it with the amount of material at the given position $s_t$ if this is higher than $S_t$ we deposit $s_t - S_t$, i.e.,

$$m_{t+1} = m_t + (s_t - S_t).$$

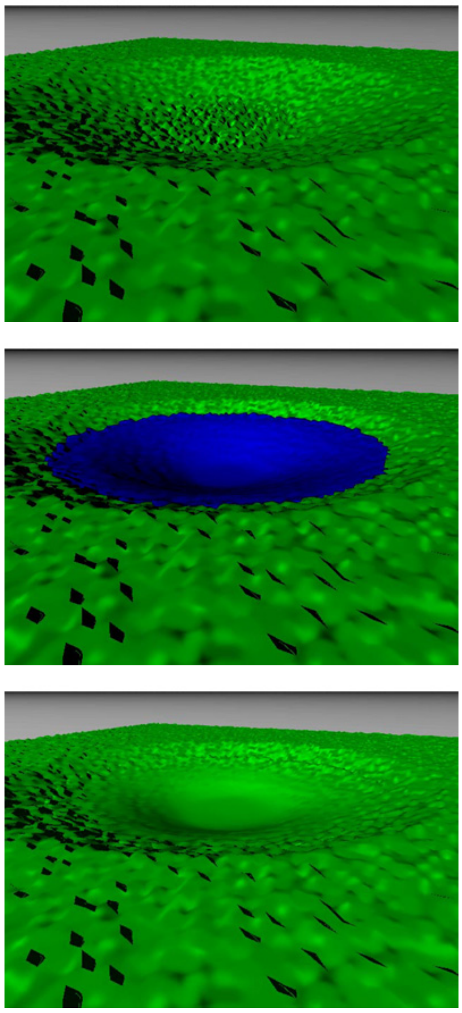
Figure 2: Process of the material deposition. An artificial pool evaporates water and the sediments are deposited at its bottom.

Figure 2 demonstrates this phenomenon. A hole (an artificial white noise surface) is filled by highly saturated water that evaporates. The bottom of the pool of water becomes flat because of the deposited material. It is important to notice that we cannot just keep the amount of soil fixed for the water volume within the vertex. The soil travels in the water and tries to reach volume equilibrium. The water and soil transport are described in the next section.

## 2.4 Water and Sediment Transport

The water motion in the nature is very complex and it can be completely described by the Navier-Stokes equations. The solution of these equations is quite complex and in the context of this has been used several times for purposes of animation [7, 8, 16]. The water transport used in the techniques previously introduced for erosion simulation is more or less precise approximation of the solution of these equations.

For the slow water motion we can apply a simple diffusion model. For each location of the height field the amount of water that exceeds the neighbors is detected and moved to the neighbors that are located bellow. For the sake of clarity let us suppose just two elements. The height that is reached by the level of the water in the actual position is the sum of all contributing quantities

$$h = m + w$$

whereas the height of the neighbor is:

$$h' = m' + w'.$$

If the difference $h - h'$ is smaller than zero, it means the actual element is in the hole, we cannot remove anything. If the element is not the local hole, we have two important cases as described in the Figure 3. The left drawings show the situation before whereas the right ones after the water transport. The upper row shows the case when just a part of water must be moved, whereas the bottom row explains the case when we should move everything.

A special care must be taken to the distribution of water in the full two-dimensional implementation. The amount of water must be

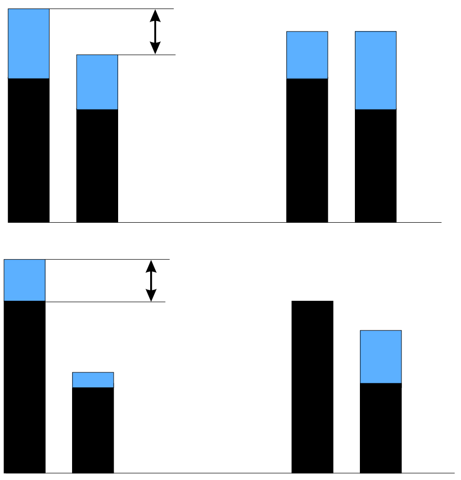
Figure 3: Two important cases for the water transportation (up and down). The left drawings shows the situation before and the right after the transport.

distributed proportionally to the relative differences of the neighboring lower vertices.

We have implemented this in the following way. Let $h_i; i = 0, 1, \ldots 7$ be the height of the neighboring vertices and $h$ the height of the vertex in the middle (see Figure 4). The

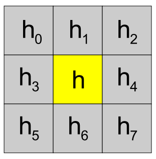

Figure 4: Indexing of the neighboring vertices.

way to achieve the correct distribution of the water is to accumulate all lower differences in a counter, let denote this counter by $sum$. The quantity of the water that should be removed $\Delta w$ is distributed according to the equation

$$\Delta w_i = \Delta w \frac{h_i}{sum},$$

where $\Delta w_i$ is the amount of water to be moved to the $i$-th vertex and the $h_i$ is the difference of the lower neighboring vertex.

An important fact not described in the previous work is that reaching equilibrium with water does not mean that the system is stable. Neighboring vertices can contain very different levels of dissolved sediments that should be equally distributed as well. Correct solution involves solving the sediment transportation the water as well. We use a solution that is an adaptation of the above described water transport. Instead of manipulating the height of the vertex we distribute the concentration of the sediments in the water.

## 3 Implementation and Results

We have implemented this technique in C and tested on IBM PC 500MHz. The program runs 500 complete erosion step of an array $400 \times 400$ elements less than two minutes. We use OpenGL for fast preview and ray-tracer Persistence of Vision for photorealistic images.

The algorithm consists of four independent steps. First we add water into the system. Next we erode the terrain i.e., we capture the material in the volume of water. Then the water and the captured material are distributed. Some part of water is evaporated and the last step is the material deposition.

The biggest advantage of the algorithm is the separation of the erosion process into these steps. We can benefit from this and apply some steps more frequently as we want to simulate certain phenomena. For example some processes are influenced by the evaporation (such as in the Figure 2), so it makes no sense to devote too much effort to the water transportation. Indeed, in this case we have applied the evaporation and deposition much more times than the water transportation. The result, although not physically exact, is realistic.

Another example, where the evaporation is not very important, but the water transportation is, is shown in Figure 5. We put three sources of highly saturated water on the non-steepy slope. As the water flows down the soil is repeatedly deposited and dissolved. This forms characteristic ridges on the slope and down at the bottom of the hill. The height field resolution was $150 \times 150$ elements and the simulation has run less than two minutes.

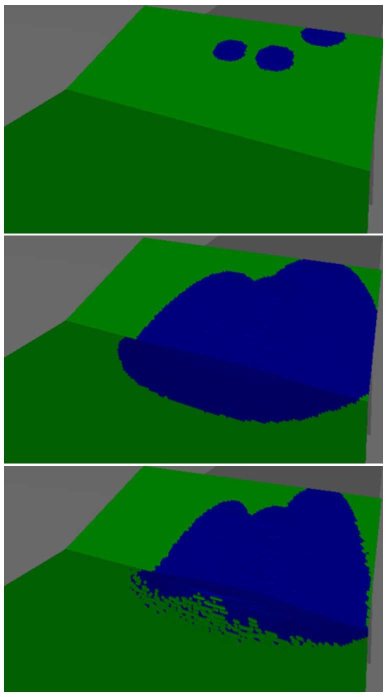

Figure 5: Three sources of highly saturated water are spilling the water that produces characteristic ridges on the hill itself as well as on its bottom.

Next example in the Figure 6 shows simulation of rain on the Olympus Mons on Mars. We have obtained the data from NASA project MOLA [13]. The volcano is completely covered by water on the first image. We were just interested in the simulation of the running water, so we have skipped the process of erosion and deposition. The program cycle consists just of rain and the water transport. Since the surface of the mountain is quite steepy, the water goes down quite fast and the erosion steps described in this paper are not affecting it very much. An interesting observation is the

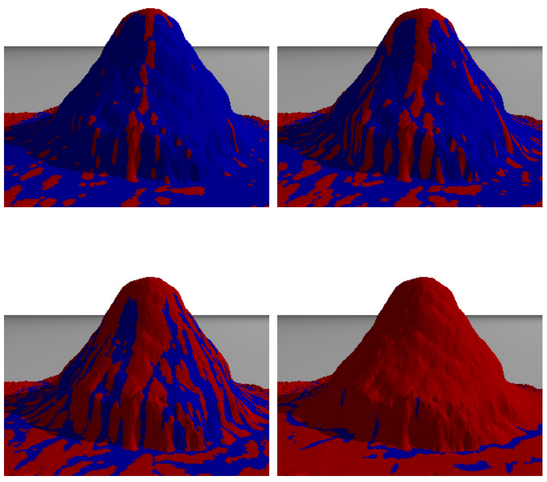

Figure 6: Simulation of the fictive rain on the martial volcano Olympus Mons.

way the water finds. The paths that the water forms are quite convincing, so we can imagine that this could be the way the water was really running.

The last example in Figure 7 shows area of Valles Marineris on Mars (approximately 1000km$\times$1000km in resolution 14km$^2$ per vertex). In this area the water supposedly had flown. We have applied heavy rain and intensive water transport (corresponding to raining season) to this area and then very dry season. During the raining season the surface was just filled of water, whereas in the dry season the water has flown in the lower areas, so we can see how the craters and valleys are filled by water that is then evaporated as well. Heavy sediment deposition also occurred during the dry season.

## 4 Conclusions and Future Work

Two important things were introduced in this paper. First, the existing algorithms were clarified and divided into four distinct and independent steps. First the water appears at some places, then it erodes underlying terrain structures, and then it transports captured material and the water deposits the material at the end. These steps run independently so we can repeat some of them in cycles in order to simulate heavy rains, or dry seasons.

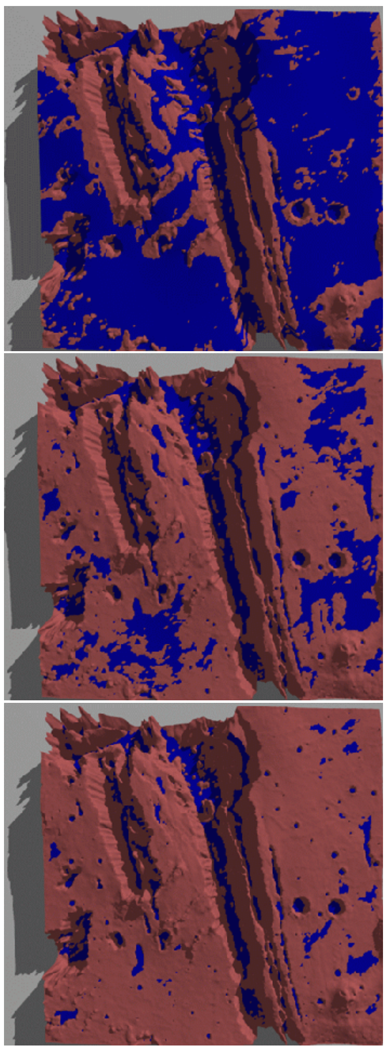

Figure 7: Drying the area of Valles Marineris on Mars after a heavy rain.

Second important thing is the new algorithm simulating hydraulic erosion that has been introduced. The algorithm is based on the fact that water sediment capacity is constant and during the evaporation this level is exceeded so some material must be deposited. The material must also be transported in the water to achieve realistic simulations.

One of the biggest weaknesses of all erosion algorithms is the algorithm that simulates the transportation of water. The possible future work involves better algorithm for the water transportation. An interesting topic is also precise simulation of the material motion in the water.

This work was supported by the ITESM grant 32419 and by the CONACyT grant 38172-A.

## REFERENCES

[1] B. Beneš and R. Forsbach. Layered Data Structure for Visual Simulation of Terrain Erosion. In *IEEE Proceedings of SCCG'01*, volume 25(4), pages 80–86, 2001.

[2] H. Chanson. *The Hydraulic of Open Channel Flow*. John Willey and Sons, 1999.

[3] N. Chiba, K. Muraoka, and K. Fujita. An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain scenery. *The Journal of Visualization and Computer Animation*, 9:185–194, 1997.

[4] A.R. Dixon and G.H. Kirby. A Data Structure for Artificial Terrain Generation. *Computer Graphics Forum*, 13:37–48, 1994.

[5] J. Dorsey, A. Edelman, H. W. Jensen, and H. K. Pedersen. Modeling and Rendering if Weathered Stone. In *Proceedings of SIGGRAPH '99*, volume 25(4) of *Annual Conference Series*, pages 225–234, 1999.

[6] D. Eberly. *3D Game Engine Design - A Pracical Approach to Real-Time Computer Graphics*. Morgan Kaufmann Publishers, 2001.

[7] N. Foster and R. Fedkiw. Practical Animation of Liquids. In *Proceedings of SIGGRAPH '01*, volume 26(4) of *Annual Conference Series*, pages 1–8, 2001.

[8] N. Foster and D. Metaxas. Practical Animation of Liquids. In *Proceedings of Graphical Models and Image Proceedings*, volume 58(5), pages 471–483, 1996.

[9] A.D. Kelley, M.C. Malin, and G.M. Nielson. Terrain Simulation Using a Model of Stream Errosion. *Computer Graphics*, 22(4):263–268, 1988.

[10] F.K. Musgrave. Towards the Synthetic Universe. *IEEE Computer Graphics and Applications*, pages 10–13, 1999.

[11] F.K. Musgrave, C.E. Kolb, and R.S Mace. The Synthesis and Rendering of Eroded Fractal Terrains. *Computer Graphics*, 23(3):11–1–11–9, 1989.

[12] K. Nagashima. Computer Generation of Eroded Valley and Mountain Terrains. *The Visual Computer*, 13:456–464, 1997.

[13] http://ltpwww.gsfc.nasa.gov/tharsis/mola.html.

[14] M. Newson. *Land, Water and Development*. Routledge, 1997.

[15] N. Ozawa and I. Fujishiro. *A Morphological Approach to Volume Synthesis of Weathered Stones*. Springer-Verlag, 2000, 2000.

[16] J. Stam. Stable Fluids. In *Proceedings of SIGGRAPH '99*, volume 25(4) of *Annual Conference Series*, pages 121–128, 1999.