4th Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2007)
J. Dingliana, F. Ganovelli (Editors)

Real-Time Erosion Using Shallow Water Simulation

Bedřich Beneš

bbenes@purdue.edu
Department of Computer Graphics technology
The Envision Center for Data Perceptualization
Purdue University, USA

Abstract

We present a new real-time hydraulic erosion simulation for Computer Graphics. In our system water runs over the surface and disintegrates the underlying layer of soil. The grit is simulated as a fluid with higher viscosity and moves on the ground of the water pool. When water evaporates, or the dissolved soil exceeds a critical level, the dissolved matter is deposited back on the ground for accumulation to occur. The grit motion as well as the water simulation are calculated by the shallow water simulation that is a 2D simplification of Navier-Stokes equations. This simulation has proven to be useful in many Computer Graphics applications because of the speed of calculation and the visual plausibility of the results. Our experiments show that the shallow water-based erosion is suitable for real-time simulation of a wide variety of phenomena including river and lake formation due to rain and evaporation, erosion of surfaces affected by a sudden splash of high level of water, mountain erosion, etc. The speed of simulation makes the algorithm suitable for real-time surface modeling and editing.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling

1. Introduction

Terrains are important visual clues that are indispensable in Computer Graphics applications and that is why terrain generation and modeling pose an important problem in Computer Graphics. Techniques for terrain geometry and topology generation range from procedural techniques [EMP*02] to conversion of data from remote sensing tools and Geographical Information Systems. A terrain generated from those techniques is usually not sufficient for a particular application and must be further edited or improved. Interactive tools and applications are typically used for computer animation, gaming purposes, etc. They provide algorithms for features smoothing, sharpening, river, mountain, or lake generation, etc. These techniques can be used in an interactive manner, where the user "paints" the desired effect on a surface. The results, although visually plausible, are barely physically correct and can lead to further problems when integrated into large systems. There is also a common agreement that physically-based techniques provide visually correct and better results. This has lead attention to physically-based terrain generation and erosion models that are usually slow and involve many input parameters.

We present a new physically-based method that simulates hydraulic erosion on 2D height-fields of sizes up to 400 $\times$ 400 elements at interactive rates on an off-the-shelf computer. An example is shown in Figure 1, where two columns of water erode a fractal terrain. The water spreads on the surface in a realistic way and leaves eroded areas behind. The first image shows the original surface and the image d) shows the final eroded surface after evaporation. Images b) and c) show the water progressing on the surface.

The principal idea of our technique is to use a shallow water equation to simulate water motion and the transportation of dissolved material. Water motion is first calculated by the shallow water equation. At the same time water dissolves a layer of material on the ground. This grit then moves as a material with higher viscosity, but only if water is present. By using this method a motion of low-layered mud and sand in water as well as fluvial erosion caused by rain are simulated. Motion of this layer is also solved by the shallow water equation again. As water moves through the terrain, it dissolves

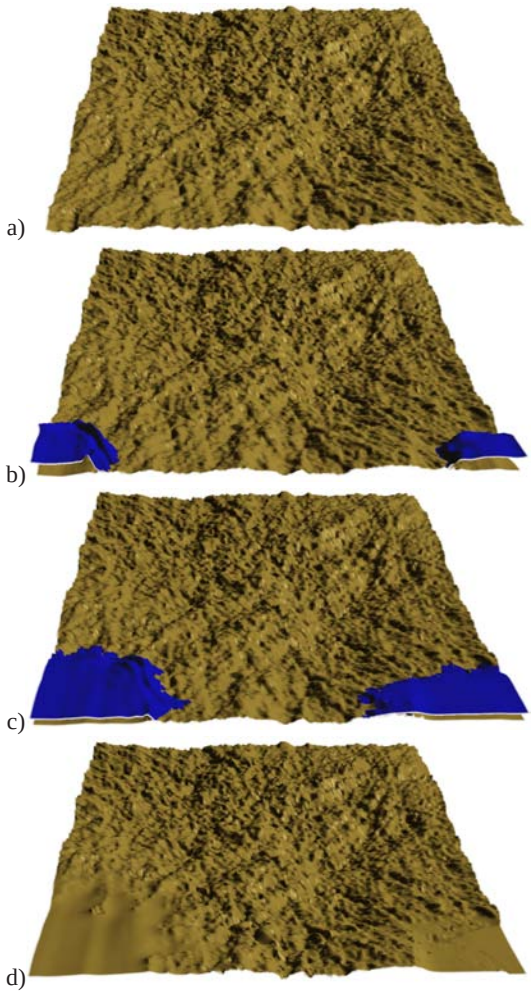

Figure 1: *An example of erosion by two columns of water suddenly splashing a fractal surface. Water trajectory is clearly smoothed by the erosion process*

underlying soil that moves as well. When water disappears or its level is smaller than a user-defined level, some material is deposited back to soil, carving the terrain features. Our method has proven to simulate a wide variety of phenomena ranging from river formation, smoothing terrains due to rain, bank and river-bed erosion, etc. The main advantage is its speed that allows, even for the non-optimized CPU version, to run medium-size terrains at interactive framerates.

2. Erosion and Geomorphological Processes

The most important morphological phenomenon is *erosion* that is a process of material transportation which smoothes a surface by moving material away. Erosion, in general, can can be described as a three step process. First some part of material is captured by an external force and disintegrated from the surface. Another force then takes the material and moves it away. In the third phase, the material is deposited to a new location. The external force that causes the erosion gives different names to different kinds of erosion. A good classification of terrain erosion for Computer Graphics can be found in the book [DL04].

Weathering is a small-scale erosion that typically affects stones and rocks creating rifts on their surfaces either in a thermal or a chemical way. *Thermal weathering* causes disintegration of the material by thermal shocks in the presence of moisture that has bigger dilatations than the rock itself. *Chemical weathering* is caused by a moisture layer on the surface and is expressed as a chemical change of the material.

Denudation is another kind of erosion process that acts typically in a planar manner. The first case of denudation is *gravity-conditioned mass movement*. An example is moving sand or gravel when it reaches an equilibrium between the gravity and its inner tension. So called *splash erosion* is caused by raindrops falling onto the surface and creating elliptical footprints in the sand. The last case of the denudation is *fluvial erosion* which is probably the most important case because of its global influence. Fluvial erosion can be described as water running down the surface and carrying forward soil particles, sand, grit, debris, etc.

The above classification describes erosion processes according to the scale of which occurs. Another classification can be made according to the main force that causes the erosion. The term *hydraulic erosion* denotes geomorphodynamics phenomena that acts as a 3D force causing rills, brooks, rivers, and lakes to carve the underlying surface and it is driven by water. *Wind erosion* causes material relocation to move through the air. Typical examples of wind erosion are sand dunes and wind ripples. *Human erosion* takes various forms, the best known is probably deforestation.

3. Erosion and Weathering Models in Computer Graphics

The Computer Graphics community has considered the important role of erosion since the first computer models of terrain were generated. In the following text we describe existing methods for terrain modifications and erosion simulation. We present them according to the above described classification and all models will be further classified according to the main force that causes them, i.e., wind or water.

Musgrave *et al* [MKM89] presented one of the first papers on erosion simulation that introduced thermal weathering and fluvial denudation. Thermal weathering works only on a local scale and the eroded parts are transported by a simple diffusion model which moves them only to the local neighbors of each vertex of the height field. This causes the speed of erosion to be very slow. Fluvial erosion introduced in the paper uses the same transportation model not

for the eroded material, but for the water that holds it. Water dissolves the underlying material until saturation is reached and relocates it when water evaporates.

This paper was later extended by many authors and the main application of this algorithm is sand and mud movement simulation. Local diffusion and material relocation can be applied to move sand particles in order reach equilibrium and, such applications can be seen in [BF02] or [SOH99]. Onoue *et al* [ON03] used a virtual sandbox where sand was manipulated interactively. Dual representation of sand was used. Sand was represented as a height field on the ground and as particles in the air when dropped.

Sand and wind erosion was addressed by the work of the same authors in [ON00]. The physically-based model for dunes and sand ripples simulation proved to be very quick in forming and animating both phenomena in real time.

All of the above mentioned techniques for sand simulation deal with 2D height fields only. A fully 3D technique that simulates sand as viscous fluid was recently presented in [ZB05].

A chemical weathering method for sandstones and rocks was described in [DEJP99]. In this paper an eroded object is modeled as a set of layers with a polygonal core and a slab of voxels on its surface. The erosion occurs in the voxel layer and is modeled by a set of differential equations. The moisture enters the material and pulls out other chemical materials by their recrystallization. This defines simulated changes in color on the surface of the object. The model also accounts for erosion of the surface that causes its smoothing. Eroded stones are rendered by Monte Carlo ray tracing with subsurface scattering.

Splash erosion simulation and rainfall erosion were presented in the work of [VPLL06]. The Soil Degradation Assessment SoDA project is a collaborative activity that involves measuring real data and their simulation and visualization.

Chiba *et al* [CMF98] introduced a physically based model of erosion that takes into account physical forces caused by moving water and their effect on the underlying terrain. This algorithm was recently extended by Neidhold and Deussen [Nei05] who showed that this erosion can be provided in real time. The main disadvantage of these methods is the *ad hoc* water motion simulation.

An ad-hoc method for rocky mountains was introduced in [IFMC03]. The main asset of this technique is that it uses a fully 3D representation for erosion. The method describes a limited set of phenomena and it focuses on a special case of rocks.

Full 3D hydraulic erosion was introduced in [BTHB06]. In this paper Navier-Stokes equations are coupled with material transportation and solved on a 3D grid. Two different kinds of material are used, cohesive and cohesionless. This method is able to simulate a wide variety of phenomena, such as receding waterfalls, river bed and river bank erosion, meander break, etc. Although this model provides high quality results the main disadvantage of the full 3D erosion algorithm is its speed, which makes it impossible to use in real-time applications today.

The previous work shows a clear tendency from ad-hoc models to models that are physically-based or at least physically-inspired. Another important property of a good erosion algorithm is its response. Interactive, or preferably real-time, techniques are the most important because they provide very quick feedback and allow for model changes and recalculation. One of the disadvantages of physical models is their dependence on constants and parameters whose influence is not often clear to the user and their effect can be sometimes very subtle or confusing.

We present a new method that addresses these issues. The shallow water simulation is a real-time technique and the main parameter of the simulation is the erosion speed that is easy to understand and easy to control. Our technique deals with 2D height fields, which is the main drawback that we exchange for the speed of the algorithm. It could be better classified as layered erosion as it uses data structures introduced in [BF01].

Very recently, in fact in parallel with this paper, another paper about hydraulic erosion using shallow equation was published [MDH07]. Their work use GPU to present hydraulic erosion at interactive framerates using concepts from [BF02, Nei05].

The paper continues with Section 4 that describes the shallow water equation and its solution in depth. Followed by Section 6 which shows examples and discusses the implementation. The last section concludes the paper and opens some questions for future work.

4. Shallow Water Equation

Fluid dynamics is fully described by the Navier-Stokes equations that capture various phenomena such as swirls, turbulence, jets of water, etc. The downside of the full 3D solution is the time complexity that is $O(n^3)$, where n is the number of grid elements used for calculation. Water simulation is a very important topic in Computer Graphics and we do not discuss the vast of papers and techniques that were recently introduced. We focus only on the shallow water simulation.

The shallow water equation introduced to Computer Graphics by Kaas and Miller in [KM90] shows a simplified solution of these equations for a 2D case with non-overlaping waves that has proven to be suitable for many Computer Graphics applications. We will follow their description in this part of the text.

The shallow water is a simplification of the Navier-Stokes equations. It is assumed that water is simulated only as a

regular 2D height field and cannot form breaking waves or another full 3D phenomena. Another assumption is that the horizontal speed of a column of water is constant. These assumptions are limiting for a full 3D simulation but work well for medium and large scale simulations where small dynamic features cannot be observed.

4.1. 1D Shallow Water

We describe the shallow water equation and its numerical solution for the purposes of the implementation only. For details on the derivation of the terms and the equation itself we refer reader to [KM90] or to [ESHD05].

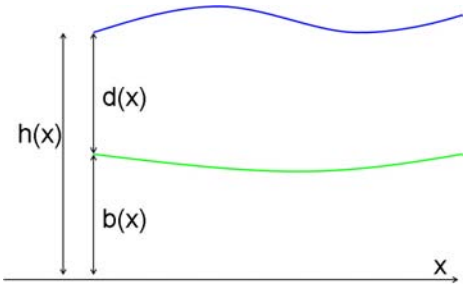

Figure 2: *Depth of water is denoted by $d(x)$, level by $h(x)$, and bottom by $b(x)$*

The shallow water equation is separable and can be described as a one dimensional case that is applied in the x and the y direction to get the full 2D solution. In the following text we suppose (see Figure 2) the level of water to be denoted by h(x), bottom by $b(x)$, and the depth of water by $d(x)$. It also holds $h(x) = d(x) + b(x)$. The continuous shallow water equation has form:

$$\frac{\partial^2 h}{\partial t^2} = gd\frac{\partial^2 h}{\partial x^2}, \tag{1}$$

where $g = 9.807[ms^{-2}]$ is the gravitational constant and the wave velocity is $\sqrt{gd}[ms^{-1}]$. This equation can be discretized into

$$\frac{\partial^2 h}{\partial t^2} = -c(d_{i-1}+d_i)(h_i - h_{i-1}) + c(d_i + d_{i+1})(h_{i+1} - h_i), \tag{2}$$

where $c = g/(2(\Delta x)^2)$ and the Δx is the space discretization of x into n discrete intervals with $i = 0, 1, \ldots, n-1$. The integration of the equation (2) leads to

$$\mathbf{A}h_i(t) = 2h_i(t-\Delta t) - h_i(t-2\Delta t). \tag{3}$$

Where $h_i(t)$ denotes an element at the position i at the time t. One needs to store the two previous solutions of the level of water in order to calculate the new one. The tridiagonal matrix $\mathbf{A}$ has the form

$$\mathbf{A} = \begin{pmatrix} e_0 & f_0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & e_1 & f_1 & 0 & 0 & 0 & 0 \\ 0 & f_1 & e_2 & \ddots & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \ddots & e_{n-3} & f_{n-3} & 0 \\ 0 & 0 & 0 & 0 & f_{n-3} & e_{n-2} & f_{n-2} \\ 0 & 0 & 0 & 0 & 0 & f_{n-2} & e_{n-1} \end{pmatrix}, \tag{4}$$

its elements e and f are

$$e_i = \begin{cases} 1 + k(d_0 + d_1) & i = 0 \\ 1 + k(d_{i-1} + 2d_i + d_{i+1}) & 0 < i < n-1 \\ 1 + k(d_{n-2} + d_{n-1}) & i = n-1 \end{cases}$$

$$f_i = -k(d_i + d_{i+1}),$$

and the $k = g(\Delta t)^2/(2(\Delta x)^2)$.

The damping of the water (viscosity) can be simulated by changing the right hand of the equation (3) as follows:

$$\mathbf{A}h_i(t) = h_i(t-\Delta t) - (1-\tau)(h_i(t-\Delta t) - h_i(t-2\Delta t)).$$

where $0 < \tau < 1$ is the damping coefficient. Usually very small values, around $\tau = 0.01$ provide reasonable damping effects.

The tridiagonal matrix $\mathbf{A}$ from the equation (3) can be solved with the time complexity $O(n)$ by a modified version of Gauss elimination method [PTVF92]. The solution of the equation (3) provides level of water at the actual time that is evaluated from the two previous states.

4.2. 2D Shallow Water

The transition from the 1D case into the 2D case is simple because the equation (3) is separable. It means that in order to obtain the full 2D solution the 1D solution must be applied successively to the rows and columns of the matrix and no special 2D version is necessary.

It is important to mention that the above presented solution is not the only possibility. There are different solutions to the shallow water problem and a good starting point is the book [ESHD05].

4.3. Issues and Problems

Even though the continuous shallow water equation (1) is volume preserving, this is not true for the discretized case (2). This subtle problem is critical, especially when dealing with small volumes of liquid. If the water level during the calculation crosses the depth level (negative water is created) the solution will automatically add it elsewhere. This is shown as a leak where the simulation will drop accumulated water at undesirable locations and present serious problems. The solution proposed in [KM90] is to analyze the

1D continuous pools of water, store the volume before and after the iteration, and redistribute the difference equally in the new pool. It does not provide realistic results and a better solution must be used. Instead of using 1D cases, we have to perform the volume preservation step in 2D. The problem reduces to finding all continuous pools in a 2D matrix, and calculating the volume difference before and after the iteration step and redistributing the pool.

5. Erosion Using Shallow Water

Erosion algorithms can be described as a three step processes: soil disintegration, material transportation, and its deposition. We build our algorithm on the following assumption: the disintegration of soil is caused by water interacting with the soil surface. As it penetrates into certain depth and mixes with the particles of soil, water creates a layer of regolith on the ground of the pool. Even when the water is still this layer will move and reach equilibrium. In moving water the bottom layer moves as well. The water motion is driven by gravity as is the regolith layer at the ground of pools. As long as the layer is fed by water it lives its own life and moves as a viscous fluid. When water evaporates or its level decreases deposition occurs. The soft layer on the ground hardens and changes back to soil.

The entire algorithm can be described by the following snippet

- Calculate the shallow water.
- Calculate the disintegration/deposition of the soil in the water.
- Calculate the shallow water applied to the soft layer of regolith.

The shallow water equation solution was described in Section 4, so we focus in this part of the text on the disintegration and deposition problems. To do this we have to extend the concept of level of water and depth in the following way (see Figure 3). As in the case of shallow water in Figure 2 depth of water is denoted by $d(x)$, level by $h(x)$, bottom by $b(x)$. The erosion related terms are regolith width $r(x)$ and the level of regolith by $br(x)$. Apparently $h(x) = b(x) + r(x) + d(x)$, $br(x) = b(x) + r(x)$ and also $h(x) = br(x) + d(x)$.

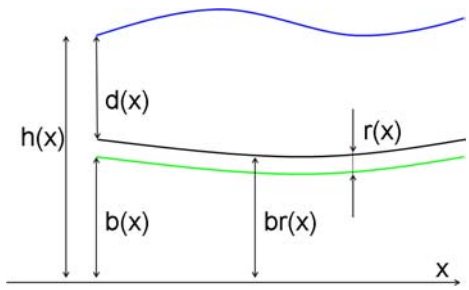

Figure 3: *Symbols used in the erosion process. Depth of water is denoted by $d(x)$, level by $h(x)$, bottom by $b(x)$, regolith width is denoted by $r(x)$ and the level of regolith by $br(x)$*

5.1. Disintegration and Deposition

During the process of disintegration the level of water penetrates the bottom $b(x)$ and changes its upper part into the regolith $r(x)$. In nature the process depends on the type of soil, the time water interacts with the soil, and the amount of water. For this to occur the simulation would; slow down, introduce more parameters, and contribute little to large scale simulations, all of which is not the focus of this text.

We define material property c_r that characterizes the material and defines how deep the soil will be eroded out, in other words c_r characterizes the highest amount of penetration and the maximum depth of the layer $r(x)$. We found useful values of $0.000 \leq c_r \leq 0.01$ in our simulations. The smaller the constant is the softer the erosion and the longer the time needed to simulate the erosion process. Large numbers can lead to unwanted oscillations in the disintegration and deposition that show as waves and ridges in the deposited material.

Keeping in mind the assumptions are oversimplifying for hydrology we assume that the level of penetration scales linearly with the amount of water up to the level of saturation c_r that is defined by user. The actual value of $r(x)$ is determined by the following equation:

$$r(x) = \begin{cases} d(x) & d(x) < c_r \\ c_r & d(x) \geq c_r \end{cases} \tag{5}$$

The response of the system is calculated as immediate. In reality the disintegration takes some time and should be calculated by eliminating some small ε of the material until the desired level is reached. In our system we consider the Δt is in order of tenths of seconds and we assume the water has enough time to penetrate into the desired level. Instead of iterating the erosion we subtract the entire level as a whole in one step.

The process of deposition is inverse to the disintegration. If the amount of dissolved material is higher than the allowed level specified by the equation (5) the excess of material is deposited and changed into the bottom $b(x)$. In other words, in every point of the surface the actual level of regolith is compared with the desired level specified by the equation (5). The difference is added or subtracted from the bottom $b(x)$ and subtracted or added to the $r(x)$.

Figure 4 shows the process of deposition and erosion taken from a 1D case of the shallow water equation. The simulation was calculated on an 1D array of 2k vertices and was running at a speed of about 2500 frames per second. The parameters of the simulation were $\Delta x = 0.1$, $\Delta t = 0.01$ and $\tau_w = 0.001$ for water and $\tau_r = 0.01$ for regolith. The

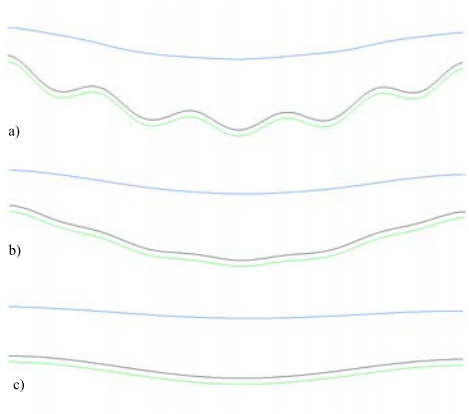

Figure 4: *An example of 1D erosion. A level of water that does not move at all has smoothing effect on the bottom*

process shows three images with the level of water, regolith, and bottom. The level of water is not moving significantly, but the regolith accumulates at the bottom and deposits because of the level of saturation. This causes smoothing of the ground.

5.2. The Algorithm

The algorithm itself consists of the three following steps. First, the shallow water is calculated for the level of water; Second, the disintegration/deposition is applied to the bottom, and lastly the shallow water equation is applied to the regolith.

Applying the simulation to water only one time and then running the disintegration/deposition step caused unwanted peaks and oscillations because water is not in a steady state after one Δt. To avoid this problem we have to bring the level of water into a steady state that typically occurs after $5-10$ iterations. An apparent adaptive solution would be to measure the velocity field of the level of water $u(x,y)$ and based on the maximum value or the speed of convergence decide if water is in a stable state or not. The entire algorithm has the following form:

1. Run the shallow water for $h(x,y)$ until water is stable.
2. Calculate disintegration/deposition and update $r(x)$ and $b(x)$.
3. Run the shallow water for $r(x,y)$ until regolith is stable.

6. Implementation and Results

The entire algorithm is easy to implement and is primarily a 1D shallow water equation that is applied to the level of water and regolith. The calculation of regolith is straightforward and easy to implement as well. We have implemented the entire algorithm in C++ and use OpenGL visualization to show the results. All experiments were run on a laptop

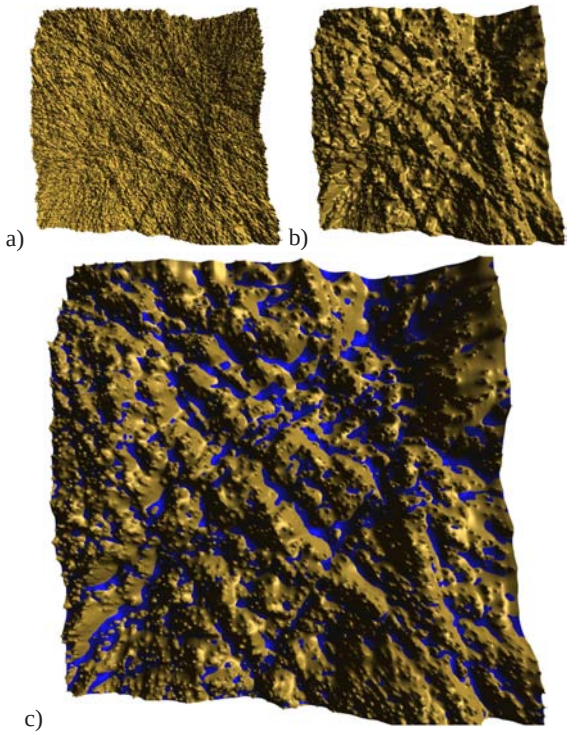

Figure 5: *An example of rivers and lakes creation. A fractal surface a) is exposed to epochs of rain and evaporations. After 10 seconds of simulation (100 steps) rivers and lakes form on the surface b) and c) and the surface is smoother in the valleys*

with Dual Core CPU running at 1.8GHz, with 2GB of memory and an Nvidia Quadro FX 2500M with 512MB of video RAM. The 1D cases of up to 2000 vertices were running at framerates of 1k fps to 3k fps. 2D simulations with height fields of 300×300 elements were running at 5-10 fps. The complexity of the algorithm is linear and depends only on the number of vertices, which is the greatest advantage of the entire technique. The solution of the equation (3) has linear time complexity and must be solved for every vertex of the height field 2x (once in the x direction and once in the y). The memory requirements are rather high. For every vertex of the height field the $h(x)$, its two copies for $h(t-\Delta t)$, and $h(t-2\Delta t)$, as well as for $r(x)$ must be stored. The level of water, as well as its depth, require only one copy in memory. The other variables can be evaluated on the fly. On the other hand since the entire information is only 2D, the requirements are much lower than of any 3D representation.

We have tested the entire system on a set of artificial experiments. In the first one a fractal surface was created by random faults method and a high frequency white noise was added to its surface. The mountain was repeatedly splashed by rain simulated as blue noise (see [Nei05]) and the wa-

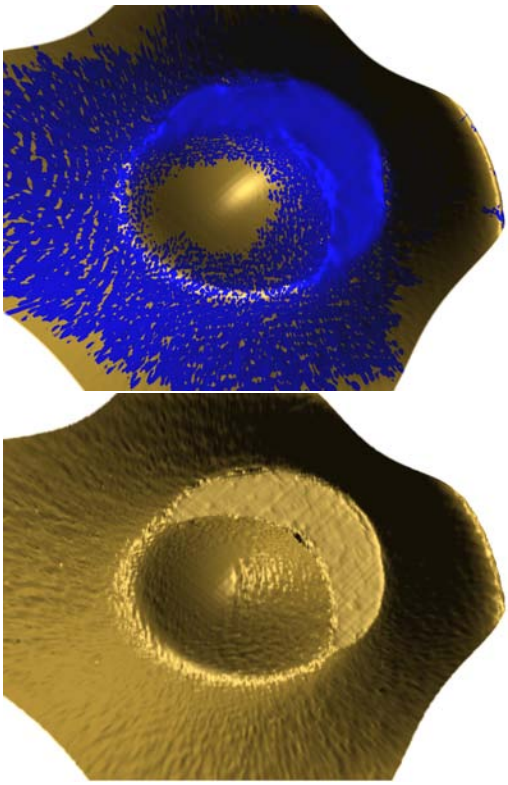

Figure 6: *An example erosion of a smooth surface. A sin-wave-like surface is eroded by seasons of rain and evaporation. A pool of deposited material clearly forms at the bottom, while the walls are eroded*

ter was then evaporated. In this way the water had enough time to run into nearby valleys and erode them out. After approximately 10 seconds of simulation of the height field at resolution 400×400, the water clearly carves rivers and lakes in the mountain as can be seen in Figure 5.

Another example in Figure 6 shows rain and evaporation that is being applied onto a smooth sine-like waved surface. After several iterations the pool of water with deposited material arises and the surface of the soil is clearly eroded. The surface size was 300×300 vertices and the speed of erosion simulation was about 5 fps.

The last example in Figure 1 shows a sudden dissolving of two water columns on a fractal surface. It is clearly visible that the erosion smoothes out the underlying terrain as water moves through it. Some other examples can be found in the accompanying video.

7. Conclusions and Future Work

We have presented an important step toward real-time deformations of terrain models in Computer Graphics using an advanced erosion model. Our system demonstrates water running over a surface and modifying the underlying soil. The grit is simulated as a fluid with high viscosity and moves on the ground of the water pool. When water evaporates, or the dissolved soil exceeds a critical level, it is deposited back on the ground and changed to soil. The grit motion as well as the water simulation are calculated by the shallow water simulation that is a 2D simplification of Navier-Stokes equations. This simulation has proven to be useful in many Computer Graphics applications because of the speed of calculation and the visual plausibility of results. Our experiments show that the algorithm is suitable for real-time simulation of a wide variety of phenomena including river and lake formation due to the rain and evaporation, erosion of surfaces affected by a sudden splash of high level of water, etc. The speed of simulation makes the algorithm suitable for real-time surface modeling and editing.

The obvious disadvantage of the presented technique is the numerous simplifications: water dissolves only the material but does not interact with it at all, one constant that describes material is good for interactive applications but is far from reality, etc. Future work should focus on clear expression of material properties and their integration into erosion models. Another step would be an integration of our algorithm into a system that allows for interactive terrain modeling. Interesting area of exploration would be an application of this erosion algorithm to surfaces of different materials.

8. Acknowledgments

We would like to thank to Matt Brisbin and Dave Whittinghill for proofreading the manuscript.

References

[BF01] BENEŠ B., FORSBACH R.: Layered Data Structure for Visual Simulation of Terrain Erosion. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics* (2001), vol. 25(4), IEEE Computer Society, pp. 80–86.

[BF02] BENEŠ B., FORSBACH R.: Visual simulation of hydraulic erosion. *Journal of WSCG 10*, 1 (2002), 79–86.

[BTHB06] BENEŠ B., TĚŠÍNSKÝ V., HORNYŠ J.,BHATIA S.: Hydraulic erosion. *Computer Animation and Virtual Worlds 17(2)* (2006), 99–108.

[CMF98] CHIBA N., MURAOKA K., FUJITA K.: An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation 9* (1998), 185–194.

[DEJP99] DORSEY J., EDELMAN A., JENSEN H. W., PEDERSEN H. K.: Modeling and Rendering of Weathered Stone. In *Proceedings of SIGGRAPH '99* (1999), vol. 25(4) of *Computer Graphics Proceedings, Annual*

Conference Series, ACM, ACM Press / ACM SIGGRAPH, pp. 225–234.

[DL04] DEUSSEN O., LINTERMANN B.: *Digital Design of Nature: Computer Generated Plants and Organics*. SpringerVerlag, 2004.

[EMP*02] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[ESHD05] ERLEBEN K., SPORRING J., HENRIKSEN K., DOHLMAN K.: *Physics-based Animation (Graphics Series)*. Charles River Media, Inc., Rockland, MA, USA, 2005.

[IFMC03] ITO T., FUJIMOTO T., MURAOKA K., CHIBA N.: Modeling rocky scenery taking into account joints. In *Computer Graphics International* (2003), pp. 244–247.

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. *Computer Graphics (Proceeding of SIGGRAPH 90) 24(4)* (1990), 49–57.

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), ACM Press, pp. 41–50.

[Nei05] NEIDHOLD, B. AND WACKER, M. AND DEUSSEN, O.: Interactive physically based Fluid and Erosion Simulation. In *Proceedings of Eurographics Workshop on Natural Phenomena* (2005), vol. 1, pp. 25–32.

[ON00] ONOUE K., NISHITA T.: A Method for Modeling and Rendering Dunes with Wind-ripples. In *Proceedings of Pacific Graphics'00* (2000), pp. 427–428.

[ON03] ONOUE K., NISHITA T.: Virtual sandbox. In *Proceedings of Pacific Graphics'03* (2003), IEEE Computer Society, pp. 252–260.

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[SOH99] SUMNER R. W., O'BRIEN J. F., HODGINS J. K.: Animating Sand, Mud, and Snow. *Computer Graphics Forum 18*, 1 (1999), 17–26.

[VPLL06] VALETTE G., PREVOST S., LUCAS L., LEONARD J.: Soda project: a simulation of soil surface degradation by rainfall. *Computers & Graphics 30*, 4 (aug 2006), 494–506.

[MDH07] MEI X., DECAUDIN P., HU B., : Fast Hydraulic Erosion Simulation and Visualization on GPU. *Pacific Graphics*, (2007), to appear.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972.