Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Interactive Terrain Modeling Using Hydraulic Erosion

Ondřej Šťava[1†] Bedřich Beneš[2] Matthew Brisbin[2] Jaroslav Křivánek[1]

[1]Czech Technical University in Prague
[2]Purdue University

Abstract

We present a step toward interactive physics-based modeling of terrains. A terrain, composed of layers of materials, is edited with interactive modeling tools built upon different physics-based erosion and deposition algorithms. First, two hydraulic erosion algorithms for running water are coupled. Areas where the motion is slow become more eroded by the dissolution erosion, whereas in the areas with faster motion, the force-based erosion prevails. Second, when the water under-erodes certain areas, slippage takes effect and the river banks fall into the water. A variety of local and global editing operation is provided. The user has a great level of control over the process and receives immediate feedback since the GPU-based erosion simulation runs at least at 20 *fps on off-the-shelf computers for scenes with grid resolution of* 2048×1024 *and four layers of material. We also present a divide and conquer approach to handle large terrain erosion, where the terrain is tiled, and each tile calculated independently on the GPU. We show a wide variety of erosion-based modeling features such as forming rivers, drying flooded areas, rain, interactive manipulation with rivers, spring, adding obstacles into the water, etc.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

1. Introduction

Even though realistic terrain modeling has been on the radar of computer graphics for more than twenty years, it still presents a challenge. Real terrains are formed by innumerable influences and modeling of all these phenomena is virtually impossible. Terrains can vary broadly in their shape and can change abruptly from place to place due to external factors such as wind, water, temperature, and chemical activities, all changing in time. The most important geomorphologic factor is certainly water, which interacts with soils in the form of hydraulic erosion.

In this work we embrace the thesis that an efficient way of modeling plausible terrains is by allowing the user to interactively apply physically-based tools emulating the aforementioned natural phenomena, particularly the hydraulic erosion. Unfortunately, interactivity of most existing physics-based methods for erosion simulation is limited and the high complexity of these algorithms only allows for the simulation of small-scale phenomena.

To address this problem, we have integrated, extended, and developed physics-based erosion algorithms that are applied in an interactive way to model terrains. The user employs our erosion algorithms to change the terrain's shape in an intuitive way with a great control over the modeling process. Since our implementation runs on the GPU at interactive frame-rates, our work is an interactive physically-based terrain modeling using erosion.

To support terrain modeling through erosion technologically, we combine two hydraulic erosion algorithms: one exploits forces and pressure fields for rapidly moving water in the manner similar to [MDH07], the other builds upon the fact that slowly moving water erodes the underlying soil by its dissolution [Ben07]. For water and soil transportation we use the pipe model described in [HW04, OH95] and recently implemented on the GPU in [MFC06]. Our algorithm simulates erosion of multiple material layers and allows for the simulation of secondary features of erosion, such as breaking the banks, or slippage, of material due to gravity. The

† e-mail: stavao1@fel.cvut.cz

© The Eurographics Association 2008.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

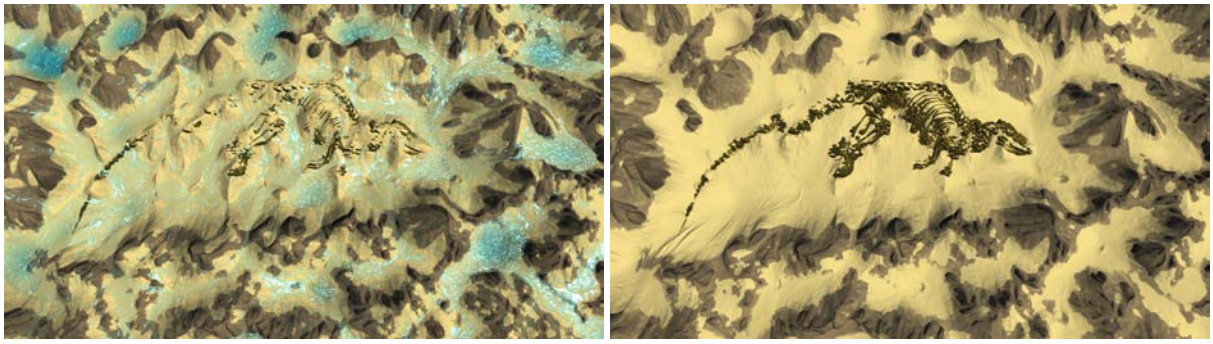

Figure 1: *Two frames from real-time simulation of erosion exposing a fossil skeleton. Falling rain erodes the upper layer and takes out particles of sand that are deposited elsewhere.*

solution is implemented fully on the GPU. To address the limited GPU memory, we divide the terrain into tiles that can be processed independently. In this way we can, for example, select a river from a large terrain and apply erosion only to the selected areas.

The results show that our interactive erosion algorithm can be used to create plausible terrains in a controlled manner, thus demonstrating the feasibility of interactive physics-based terrain modeling with immediate feedback. Figure 1 shows an example of our results.

The technical contributions of our work are: (1) integration of three erosion algorithms, (2) their extension to a layered terrain representation, (3) adaptation of the existing algorithm [Ben07] to use the pipe-model for water transportation [MDH07], (4) implementation of the algorithms on the GPU, (4) selective erosion of sub-tiles on the GPU.

2. Related Work

Terrain modeling has been a topic of computer graphics since the very beginning. First purely *procedural models* were based on fractals [Lew87, Man83], multifractals [EMP*03], and hypertexture [PH89]. The main drawback of these techniques—limited user control—has been addressed by erosion simulation or by changing local properties of fractal surfaces. Kelley et al. [KMN88] use fractal interpolation to connect predefined ridges and valleys of a terrain. Prusinkiewicz and Hammel [PH93] use L-systems [PL90] for adaptive subdivision that results in a fractal mountain with a river. A method for deforming and clipping fractal terrain to a desired shape is described in [SS05]. A step toward interactivity of procedural models is example-based modeling [BSS06, ZSTR07], where a terrain is generated from constraints, such as a 2D sketch.

On the opposite side from the procedural methods are *interactive techniques*, typically integrated into modeling software packages or provided as stand-alone applications [DP07, Ter07].

Solutions exist for modifying terrains by *physics-based techniques*. One of them was introduced in [MKM89]. Here the soil transportation occurs by gravitational water diffusion and thermal weathering. An approach for force-based erosion of running water was described in [CMF98] and recently improved in [NWD05] to work at interactive framerates for small scenes. Eroded valleys were simulated by an ad-hoc solution in [Nag97]. Hydraulic erosion is decomposed into a sequence of independent steps (rain, erode, deposit, evaporate) in [BF02]. A full 3D erosion simulation using Navier-Stokes equations [BTHB06] can simulate a wide variety of phenomena, such as concavities or receding waterfalls, but the time complexity is very high. Landscape synthesis by erosion and deposition was presented in [RPP93]. Mei et al. [MDH07] and Anh et al. [ASA07] implement real-time hydraulic erosion on the GPU. Weathering algorithms for general 3D shapes, rather than terrains, also exist. Dorsey et al. [DEJP99] simulate erosion of weathered stones; Chen et al. [CXW*05] use γ-ton tracing to simulate rust and dust. Recently Wojtan et al. [WCMT07] simulate corrosion and erosion using a chemical approach.

Our solution uses shallow water simulation [MDH07] described in more detail in Section 4. For an overview of fluid simulation in computer graphics, we refer to [BFMF06].

3. Scene Description

Terrains can be described as regular height fields [MKM89], a data structure resulting from remote sensing used in GIS as Digital Elevation Models. Although efficient for erosion simulation and rendering, height fields do not support different kinds of materials. Instead, our system uses layered representation introduced in [BF01], where each layer represents a different material and the layers are summed up for the final elevation. Such data structure can be easily created or modified by any image editing software, and furthermore, is suitable for GPU implementation. Figure 2 shows the input images used for the scene in Figure 1. Figure 3 shows a terrain exposing its defining layers and the effect of erosion

© The Eurographics Association 2008.

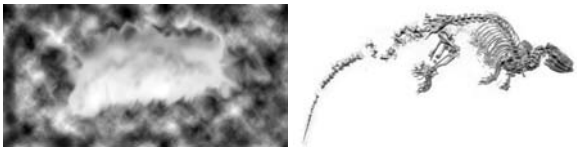

Figure 2: *Images of layers that form the scene in Figure 1.*

that adds material only to the topmost (sediment) level. Each layer is composed of a material with different erosion capabilities, such as dissolution traits or resistance to the water movement. Thanks to that, we can simulate a wide range of natural phenomena like eroding stone into sand or moving mud over a harder subsoil.

The layers can be exported as images during the simulation and imported as height fields into a third-party 3D package for further modeling and rendering. Figures 6 and 9 were rendered in Mental Ray using this approach. (Other images were rendered on the GPU as described in Section 6.)

The erosion and deposition are simulated as material exchange between the topmost layers (see Figure 3).

Another part of the scene description is the water source(s), water outlet(s), and environmental properties such as the amount of rain and the evaporation intensity. All these parameters can be controlled locally or globally by the user during the simulation. The user can also add and take out material as well as water. These interactive features are demonstrated in the accompanying video.

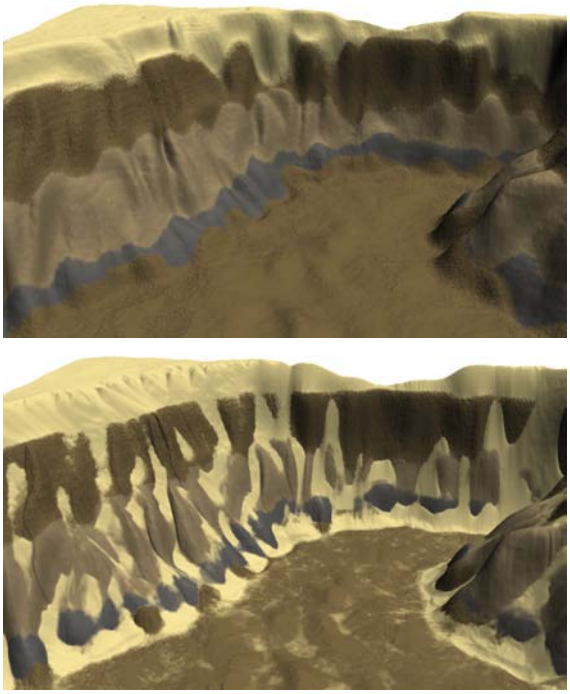

Figure 3: *A scene composed of multiple layers of material with different properties (left) is eroded. The eroded material is always deposited as the topmost layer.*

4. Water Simulation

The fundamental part of our system is a simulation of moving water. Virtually any of the existing 3D fluid dynamics methods [BFMF06] could be used, however, they typically provide non-interactive results [BTHB06]. Since our focus is on large-scale erosion, a good candidate are shallow-water models [KM90]. These were recently significantly improved and used for simulation of fluids running over terrains [HW04, MFC06], splashing fluids [OH95], and even for real-time hydraulic erosion simulation [MDH07].

For the fluid simulation and erosion, we adopt the hydrostatic pipe-model (also called the column-based model) [HW04, MFC06, MDH07, OH95] that discretizes the water volume into vertical columns with constant physical properties (see Figure 4). Columns have constant area and communicate with their four neighbors by *virtual pipes*. Using the pipes they attempt to stabilize hydrostatic pressure caused by either different levels of water in each column or by external forces. This involves water volume exchanges that, in effect, present themselves as a change in water level. This approach is inherently parallelizable and, as such, well suited for a GPU implementation [MDH07]. In the following text we briefly describe the pipe model; we refer to [HW04, MFC06, MDH07, OH95] for a more detailed description.

Let us denote the offset of the coordinates between two neighboring columns by the lower index $i, j \in \{-1, 1\}$. The difference of the static pressure $\Delta P_{i,j}$ between a column at (x,y) and its neighboring column at $(x+i, y+j)$ is then

$$\Delta P_{i,j}(x,y) = \rho g \, \Delta h_{i,j}(x,y),$$

where ρ is the fluid density, g is the gravitation acceleration, and $\Delta h_{i,j}(x,y)$ is the difference of heights between the two

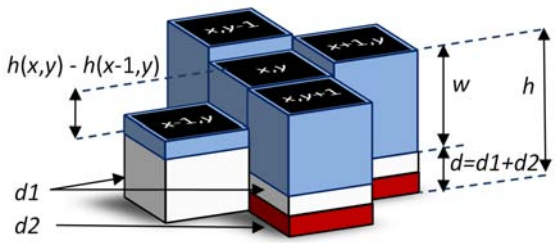

Figure 4: *Scene discretization for the pipe model. (Visible) cells $(x+1,y)$ and $(x,y-1)$ are composed of two layers (d_1 and d_2) and the cell $(x-1,y)$ only of d_1.*

© The Eurographics Association 2008.

neighboring columns. The height of a column is

$$h(x,y) = w(x,y) + \sum_k d_k(x,y),$$

where d_k are the heights of individual terrain layers and w is the height of water in the given column.

A virtual pipe tries to stabilize pressures at both ends, which results in an acceleration of a fluid flowing through this pipe. The acceleration $a_{i,j}(x,y)$ from a column at (x,y) to a column at $(x+i, y+j)$ is then given by:

$$a_{i,j}(x,y) = \frac{\Delta P_{i,j}(x,y)}{\rho l},$$

where l is the pipe length, which corresponds to the distance between two columns. We set $l = 1$ [m] in our simulations.

The simulation of the water movement is performed in discrete time steps Δt. Assuming constant acceleration over Δt, we can describe the flow in a pipe as

$$f_{i,j}^{t+\Delta t}(x,y) = f_{i,j}^{t}(x,y) + \Delta t\, C\, a_{i,j}(x,y), \tag{1}$$

where C is the cross-sectional area of the pipe. In our simulation this area is always constant with profile $C = l^2$. Finally, the new water height in each cell is:

$$w^{t+\Delta t}(x,y) = w^t(x,y) + \frac{\Delta t}{l^2} \sum_{i,j} [f_{i,j}^t(x-i, y-j) - f_{i,j}^t(x,y)].$$

During the simulation a column can eventually end up with a negative water volume. A GPU-oriented solution [MDH07] for this problem is to compute only flows with the positive velocity (output flows) in every cell. Then, we describe the total output volume of water in a given cell for a time-step Δt as:

$$V_{out}(x,y) = \Delta t \sum_{i,j} f_{i,j}(x,y),$$

and solve the problem by scaling down the output flows of the given cell if the total output volume is larger than the amount of water in the column:

$$f_{i,j}(x,y) = \begin{cases} f_{i,j}(x,y) & V_{out}(x,y) \le l^2 w(x,y) \\ \dfrac{l^2 w(x,y)}{V_{out}(x,y)} f_{i,j}(x,y) & V_{out}(x,y) > l^2 w(x,y) \end{cases}$$

Boundaries are treated as *free-slip* which means that the water velocity vector $\mathbf{v}$ is always set to zero in the direction of the normal vector $\mathbf{n}$ of the boundary so that $\mathbf{v} \cdot \mathbf{n} = 0$. This can be achieved by cloning values of water and terrain height to boundary cells from their neighboring cells.

5. Erosion algorithms

The system builds on three different erosion algorithms: force-based and dissolution-based hydraulic erosion, and direct material transportation through sediment slippage.

The force-based hydraulic erosion algorithm is based on the forces that are caused by the running water on the terrain surface. This algorithm was first introduced by [CMF98], then revised by [NWD05], and recently implemented on the GPU in [MDH07]. We extend this algorithm by an unconditionally stable solution with the second-order accuracy called semi-Lagrangian MacCormack method [SFK*07]. This erosion algorithm is well-suited for the simulation of moving water such as fast rivers or rain.

The dissolution-based hydraulic erosion algorithm is inspired by [Ben07]. The key observation is that when water dissolves the bottom of a pool or a river, the motion of the regolith in the water can be approximated as the motion of a highly viscous fluid. This kind of hydraulic erosion simulates slowly moving water volumes, has mostly smoothing effects on the bottom of the pools, and also causes bank erosion. We present two extensions of the previous work. First, [Ben07] uses a different algorithm for water simulation [KM90], whereas we integrate erosion with the pipe model. Second, we introduce a GPU implementation of this algorithm.

Thermal weathering [MKM89] is caused by thermal shocks where some part of material is crumbled by changes of temperature and deposited. When considering only regular height fields, this motion efficiently simulates sand settling [SOH99,ON00,ON03]. We have adapted this erosion to simulate the effect of soil slippage that smoothes the edges of eroded parts of the terrain. The banks can collapse by a sudden break or by a slow erosion. However, once a block of the bank falls into the water it will be dissolved in a slow manner and the effect of smoothing will prevail.

One of the main contributions of this paper is an integration of several erosion algorithms in a unified way on a multilayered data-structure. In the following text we describe the individual erosion algorithms in detail.

5.1. Force-based erosion

The force-based erosion algorithm uses the forces caused by the running water and their effect on the terrain (see Figure 5). We have enhanced the technique described in [MDH07] by a more accurate sediment transport of [SFK*07] and the support for multi-layer setting.

The key observation is that the moving mass of water causes loosening of terrain particles into the water in the form of sediment. Mei et al. [MDH07] define the *sediment transport capacity* of the flow, $S_k^m(x,y)$, as:

$$S_k^m(x,y) = \|\mathbf{v}(x,y)\|\, C_k \sin\alpha(x,y), \tag{2}$$

where C_k is the *sediment capacity constant* defined for each layer k individually, $\alpha(x,y)$ is the tilt angle of the terrain and $\mathbf{v}(x,y)$ is the water velocity. Usual values of C_k used in our simulations are between 0.0001 for rock and 0.1 for sand.

To obtain the velocity $\mathbf{v}(x,y)$, we compute the amount of water passing through a column per unit time in a specific

© The Eurographics Association 2008.

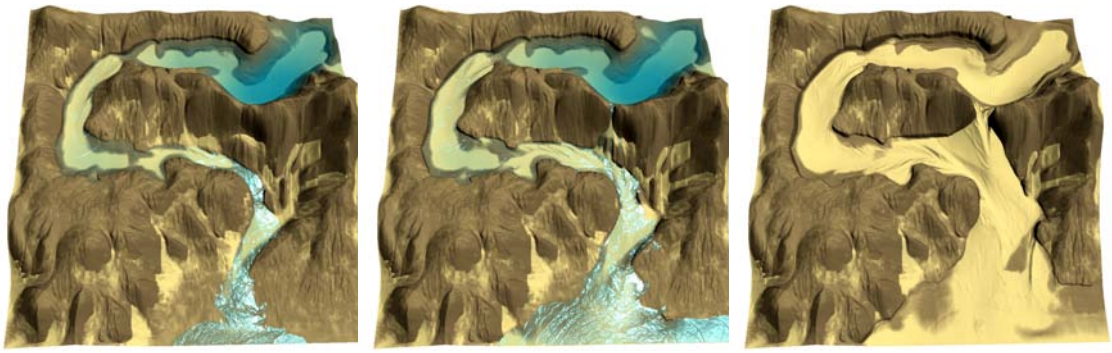

Figure 5: *Force-based erosion is suitable for erosion by running water. A meander break is shown in the image sequence.*

direction. For the x direction the amount of water is

$$W_x(x,y) = \frac{1}{2}[\, f_{1,0}(x-1,y) - f_{-1,0}(x,y) + f_{1,0}(x,y) - f_{-1,0}(x+1,y)].$$

The x-component of the velocity is:

$$v_x(x,y) = W_x(x,y) \;/\; l\bar{w}(x,y),$$

where

$$\bar{w}(x,y) = \frac{1}{2}[w^t(x,y) + w^{t-\Delta t}(x,y)]$$

is the average water height in the column during the last two steps. The y-component is computed in similar way.

Once $S_k^m(x,y)$ is established for all columns, we compare an actual level of sediment $S^a(x,y)$ with $S_k^m(x,y)$. If the actual level is larger, some sediment is deposited onto the terrain, otherwise some sediment is dissolved into water. The speed of these transformations is affected by deposition and dissolving constants [MDH07].

After the final amount of sediment in each column is calculated, the sediment is transported by the water as described by the advection equation:

$$\frac{\partial S^a}{\partial t} = -(\mathbf{v} \cdot \nabla) S^a. \tag{3}$$

Unlike [MDH07], we solve this equation with a second-order accuracy using the semi-Lagrangian MacCormack method [SFK*07], yielding reduced numerical diffusion of sediment transport.

Our extension to multi-layer setup assumes that the sediment consists only of the material in the topmost layer and that it is always deposited into the topmost layer. Therefore, we can only check if $S^a(x,y) > S_{top}^m(x,y)$ to determine deposition or dissolution. Deposition only affects the topmost layer, but each layer can participate in the dissolution step since there may not be enough material in any one layer k to fully saturate the water when $S^a(x,y) < S_k^m(x,y)$. We solve this problem by processing all layers in top-to-bottom order and subtracting the total height of layers above layer k from the maximum sediment transport capacity of that layer, yielding:

$$\hat{S}_k^m(x,y) = S_k^m(x,y) - \sum_{i=k+1}^{top} d_i.$$

The dissolution is performed only when $\hat{S}_k^m(x,y) > S^a(x,y)$. The whole multi-layer process for force-based erosion can be summarized by the following pseudocode:

```
if S^a(x,y) > S^m_top(x,y) then
  | PerformDeposition();
else
  | d_{top+1} ← 0;
  | for k ← top to 0 do
  |   | Ŝ^m_k(x,y) = S^m_k(x,y) − Σ_{i=k+1}^{top} d_i;
  |   | if Ŝ^m_k(x,y) > S^a(x,y) then
  |   |   | PerformDissolution();
  |   | else
  |   |   | break;
  |   | end
  | end
end
```

5.2. Dissolution-based erosion

The dissolution-based algorithm is based on an observation that a slowly moving water penetrates the underlying soil and creates a layer of slowly moving regolith that accumulates on the bottom until it reaches an equilibrium. As long as the layer is fed by water it remains liquid. When the water evaporates or its level decreases, deposition occurs—the soft layer hardens and changes back to soil.

The original algorithm of [Ben07] uses a CPU implementation of the shallow-water simulation as described in [KM90]. We have implemented the algorithm using the pipe-model described in Section 4 on the GPU.

Let us recall (see Figure 4) that the bottom of the column is denoted by $d(x,y)$, the depth of water by $w(x,y)$ and total

© The Eurographics Association 2008.

column height by $h(x,y)$. A regolith layer on the bottom of the water is introduced, with the thickness denoted by $r(x,y)$.

Water penetrates the bottom and changes its upper part into the regolith. The maximum penetration depth for a layer k is a material constant c_k. We found useful values of $0.0001 \leq c_k \leq 0.01$ in our simulations. The smaller the constant, the softer the erosion and the slower its progress. We assume that the maximum level of penetration scales linearly with the amount of water $w(x,y)$ up to c_k. The maximum regolith thickness is then computed as:

$$r_k^m(x,y) = \begin{cases} w(x,y) & w(x,y) < c_k \\ c_k & w(x,y) \geq c_k \end{cases} \tag{4}$$

If the current regolith thickness $r(x,y)$ exceeds the maximum $r_k^m(x,y)$ (typically on steep slopes or because of evaporation) the excess material is deposited into the topmost soil layer, otherwise some material is dissolved. At any rate, current regolith thickness is set to the maximum $r(x,y) \leftarrow r_k^m(x,y)$ and its height is updated using the shallow-water model to establish the regolith thickness for the next time-step of the simulation.

Multiple layers are handled in a similar manner to the force-based erosion. We only substitute $S_k^m(x,y)$ with $r_k^m(x,y)$ and $S^a(x,y)$ with $r(x,y)$.

Figure 6 shows the effect of dissolution erosion: the rugged terrain was smoothed out by slowly moving water and the river bank got steeper.

5.3. Material Slippage

Once exposed by erosion or otherwise deposited, sand and soil do not stay in the same position and eventually slip down by gravity. However, there is an internal tension of the material that prevents a continuous falling. This can be roughly characterized by the so called *talus angle* [MKM89]. This material property can be set as a parameter; for dry sand we take the talus angle of 30°.

Simulation of this kind of erosion involves comparing the gradient at every location to the material's talus angles. If the talus angle is exceeded, certain amount of material is removed to the locations below. We compute this amount by the following formula:

$$\Delta d = \begin{cases} \Delta t(\Delta h - l \tan\alpha) & \Delta h > l \tan\alpha \\ 0 & \Delta h \leq l \tan\alpha \end{cases} \tag{5}$$

where Δh is the difference of height between two neighboring cells and α is the talus angle.

This technique has been used for sand manipulation [LM93, GCD*98, SOH99, ON00, ON03] but we are not aware of any GPU implementation.

In a multi-layer setup, we need to consider that when the slippage of a layer is calculated, all layers above the processed layer are also affected. Therefore, we perform slippage on the layers sequentially in the bottom-to-top order.

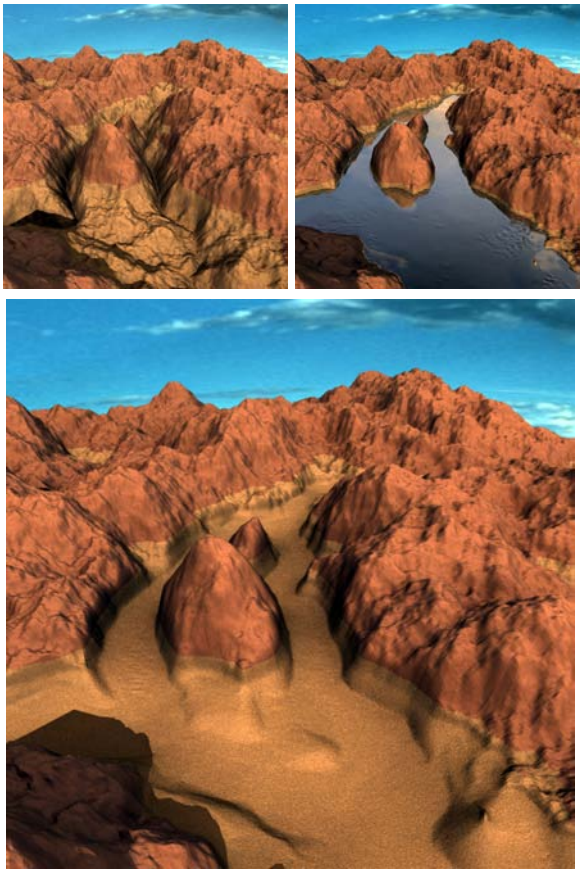

Figure 6: *Dissolution-based erosion softens the river bed and erodes the banks (Rendered using Mental Ray.)*

5.4. Material Changes

As described in Section 3, the scene is composed of multiple layers of different materials with different erosion properties (see in Figure 3). Erosion runs only on the exposed layer, which can be virtually anyone, as long as there are layers with holes above it. The topmost layer consists of light soil or sediment.

The deposition process has two steps. The sediment deposition into the highest layer and the sediment change into harder material to the layer just below. When the sediment is located without changes for a long time, it accumulates onto the lower layer and becomes eventually part of it. As the underlying layer can be virtually made of any material, we define the aging coefficient as proportional to the layer hardness. The implementation simply involves keeping a time stamp for each column of the sediment that is incremented with every simulation step. When the column is touched by water or slippage erosion, its counter is reset. This intuitive approach, although not geologically justified, results in visually plausible results.

© The Eurographics Association 2008.

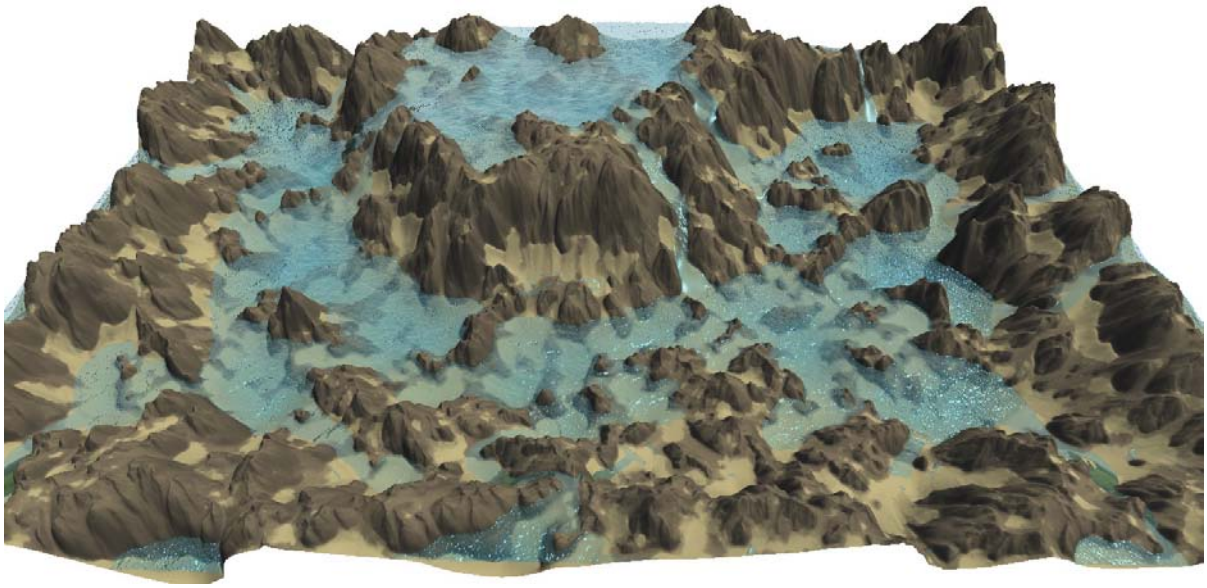

Figure 7: *Terrain eroded by the three erosions algorithms and rendered on the GPU.*

5.5. Integration

The above described algorithms are applied in the following order. First the water inputs add water into the system. It can be because of rain, user interaction, or user defined water sources. The water simulation using the pipe model is calculated and then the force-based and the dissolution-based erosion steps are evaluated. Both processes involve erosion/deposition, so as a result a terrain can exceed the talus angle. So the last step is the sediment slippage calculation and then the results are rendered.

6. Rendering

Two methods for terrain visualization were used: real-time rendering on the GPU for interactive feedback (Figures 1, 3, 5, 7, and 8) and off-line rendering in Mental Ray for high-quality results (Figures 6 and 9). In the latter case, lighting was provided by one area light source as well as using global illumination, ambient occlusion, and caustics.

For the GPU-based rendering, we displace a predefined vertex grid by the terrain and water height in the vertex shader. Normals for each grid are calculated in the fragment shader for every frame. Rendering of multiple material layers is handled by defining a thickness threshold for each material after which a layer of that material is considered fully opaque. If the actual layer thickness is less than the threshold, a proportional fraction of the material color is added to the final color at the given location. This process is performed for all layers in the top-to-bottom order. A typical example of the result is in Figure 7.

Water transparency is determined by the difference of depth values of a terrain fragment and the associated water fragment. The surface of the water reflects an environment map with the amount of reflection determined by the Fresnel term. To limit z-fighting between water and terrain levels, we discard all water fragments whose distance from the terrain surface is below a threshold value. We use the value of $0.1[\mathrm{m}]$; ideally the threshold should be determined by the terrain size and the camera settings.

7. Implementation and Results

The design of the algorithms and the simplifications made were purposely done to facilitate a GPU implementation. All simulation methods presented are executed entirely on the GPU without any data transfer from GPU to CPU during the simulation.

We use C++, OpenGL, and the NVIDIA Cg shading language. Our examples were simulated on an AMD Athlon X2 6000+ PC with an NVIDIA GeForce 8800 GTX GPU. The visualization was computed either on the same system with our own rendering engine or we exported out simulation data and rendered the images in Autodesk Maya 8.5 Mental Ray. No geometry was further edited in order to show the exact results of the modeling process.

In this section, we sketch our GPU implementation; an in-depth description can be found in [ŠBK08]. The GPU implementation first defines the computational domain by rendering a quad with screen-space dimensions equal to the resolution of the computational grid. The rasterization unit then generates fragments, each corresponding to one column in

© The Eurographics Association 2008.

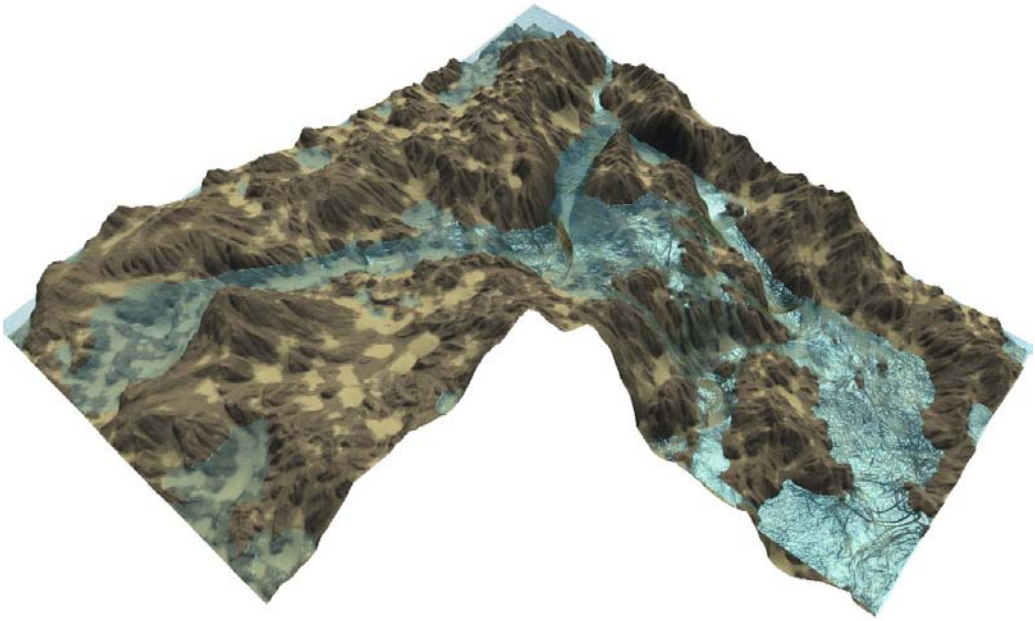

Figure 8: *Terrain is divided into tiles that are calculated independently and synchronized at borders.*

the virtual-pipe model, and launches the fragment shaders that perform one time-step of the simulation. Individual data arrays, like velocity field or water height, are represented by 32-bit floating point textures. Experiments have shown that 16-bit precision may results in serious issues with water flow computation.

Multiple material layers are handled by packing each layer into one component of a related texture. Although more layers can be stored in different textures, the maximum of four layers allows for a simulation of a wide variety of phenomena and was not a limitation in the production of our examples. The layers are processed by iterating in the top-to-bottom order. Since some results of multi-layer computations are stored in dynamically indexed arrays, the simulation is currently limited to GPUs with Shader Model 4 capabilities as earlier generations do not support this feature.

Boundary conditions are implemented using so called ghost cells [Fed03] that contain a copy of their neighbors and act as a free-slip boundary where no water can run out of the simulation domain. However, water movement along the border is not restricted.

Our implementation supports simulation of multiple terrain grids (tiles) individually. This feature enables us to simulate non-rectangular terrains efficiently (see Figure 8 and the accompanying video) and has the potential to be used in multi-GPU systems. The synchronization of neighboring tiles at their borders is performed by copying of border values before each simulation step. Tile dimensions and computational domain are extended to handle these copied values.

Figure 7 shows a fractal terrain affected by rain, where only global user control is used. The mesh resolution was 2048×1024 and four layers of material were used. All three erosion algorithms work together as they erode and smooth out the terrain features. The user is free to control the process, can manipulate the rain, zoom to see details, stop/resume the erosion, and eventually save the scene for later work. The fossil skeleton scene in Figure 1 is another example of a rain-eroded terrain.

Figure 9 shows the effect of a local editing. A flooding wave is manually created and the water runs down the terrain and erodes small hills. The scene was rendered using Mental Ray.

More results can be seen in the accompanying video that presents interactive editing of the scene, water sources and sinks manipulation, interactive terrain layer editing, and individual effects of all three erosion algorithms as well as the algorithms working together. The purpose of this paper is to show the *feasibility* of this modeling approach. However, user studies are required to assess its actual usefulness.

Simulation times summarized in Table 1 shows a linear growth of the processing time as the function of the grid resolution. There is a penalty for using multi-layer terrain setup due to the use of four-component textures for all multi-layer terrains. It is worth noting that the real-time rendering usually takes more time than the simulation of the erosion itself.

© The Eurographics Association 2008.

Grid size ⇓	Fluid sim. [ms]			Vel.-based ero. [ms]			Diss.-based ero. [ms]			Slippage [ms]			Total [ms]		
Layers ⇒	1	2	4	1	2	4	1	2	4	1	2	4	1	2	4
256x256	0.19	0.3	0.31	0.67	0.74	0.75	0.2	0.34	0.34	0.14	0.39	0.75	1.22	1.78	2.1
512x512	0.57	1.0	1.05	2.0	2.35	2.38	0.64	1.16	1.16	0.46	1.35	2.73	3.72	4.9	7.3
1024x1024	2.18	4.13	4.15	7.84	9.09	9.15	2.4	4.48	4.49	1.8	5.45	10.8	14.26	23.20	28.7
2048x1024	4.03	7.74	7.75	14.53	17.22	17.24	4.54	8.62	8.67	3.38	10.0	20.1	26.54	43.92	54.4

Table 1: *Performance of the algorithm as a function of the number of layers and the simulation grid resolution.*

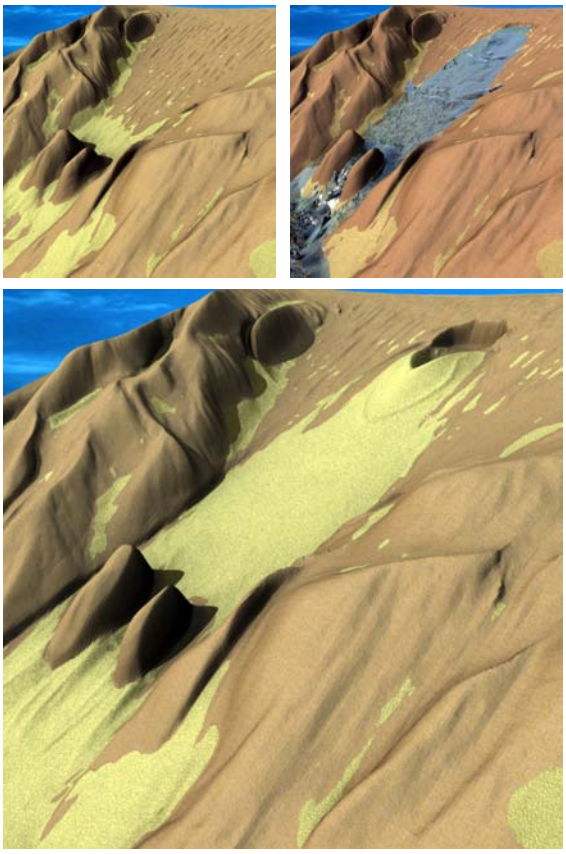

Figure 9: *A flooded valley. Images rendered in Mental Ray clearly show the erosion/deposition effects.*

The most limiting factor of the implementation is the memory consumption of the GPU solver. The use of 32-bit floating point textures limits maximum dimensions of the simulation grid to 2048×1024 for a four-layer scene on today's GPUs.

8. Conclusion and Future Work

We have presented a set of physically-based erosion tools for real-time interactive terrain modeling. A terrain is represented as a layered height-field with set of intuitive erosion related parameters that can be changed interactively. Variety of local and global control over the modeling is provided by editing operations, such as add spring, dry, evaporate, rain, add obstacle of easy-to-erode material, etc. Two hydraulic erosion algorithms for running water are coupled. Areas where the motion is slow are eroded by the dissolution erosion, whereas in the areas where water motion is faster the force-based erosion prevails. When the water under-erodes certain areas sediment slippage takes effect and the river banks are eroded out. The user has a great level of control over the terrain formation process and the GPU-based erosion algorithm runs at 20 fps on off-the-shelf computers for scenes with simulation grid resolution 2048×1024 pixels and four layers of material. We also present a tile-based solution for large terrain erosion, where the terrain is tiled and each tile is calculated independently on the GPU. We show a wide variety of erosion-based modeling effect such as forming rivers, drying flooded areas, rain, interactive manipulation with rivers, spring, adding obstacles into the water.

Future work could address the limited grid resolution either by running simulation for individual tiles on separate GPUs or by using an adaptive grid such as [LKS*05]. Furthermore, a full 3D erosion simulation would greatly enhance terrain modeling possibilities. To that end, we are considering to couple erosion with smoothed particle hydrodynamics. Our goal is erosion simulation that runs in real-time and results in visually plausible terrain shapes. To this end, the underlying models as well as the computational methods used, although physically-based, are just approximate, and therefore cannot be predictive of real-world situations.

Acknowledgements

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics) and the Aktion Kontakt grant no. 48p11.

References

[ASA07] ANH N. H., SOURIN A., ASWANI P.: Physically based hydraulic erosion simulation on graphics processing unit. In *Proc. of GRAPHITE '07* (2007).

[Ben07] BENEŠ B.: Real-time erosion using shallow water simulation. In *VRIPHYS '07: Proc. of Workshop in Virtual Reality Interactions and Physical Simulation* (2007).

[BF01] BENEŠ B., FORSBACH R.: Layered data representation for visual simulation of terrain erosion. In *SCCG '01: Proc. of the 17th Spring conference on Computer graphics* (2001), p. 80.

© The Eurographics Association 2008.

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

[BFMF06] BRIDSON R., FEDKIW R., MULLER-FISCHER M.: Fluid simulation. In *SIGGRAPH 2006 Course Notes #14* (2006).

[BSS06] BROSZ J., SAMAVATI F. F., SOUSA M. C.: Terrain synthesis by-example. In *Advances in Computer Graphics and Computer Vision* (2006), pp. 122–133.

[BTHB06] BENEŠ B., TĚŠÍNSKÝ V., HORNYŠ J., BHATIA S. K.: 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*, 4 (1998), 185–194.

[CXW*05] CHEN Y., XIA L., WONG T.-T., TONG X., BAO H., GUO B., SHUM H.-Y.: Visual simulation of weathering by γ-ton tracing. *ACM Trans. Graph. 24*, 3 (2005), 1127–1133.

[DEJP99] DORSEY J., EDELMAN A., JENSEN H. W., PEDERSEN H. K.: Modeling and rendering of weathered stone. In *Proc. of SIGGRAPH '99* (1999), pp. 225–234.

[DP07] DAZ-PRODUCTION: Bryce 6. http://www.daz3d.com/i.x/software/bryce/, 2007.

[EMP*03] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling*, 3rd ed. Academic Press, Inc., 2003.

[Fed03] FEDKIW R.: Simulating natural phenomena for computer graphics. In *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer Verlag, New York, 2003.

[GCD*98] GASCUEL J.-D., CANI M.-P., DESBRUN M., LEROY E., MIRGON C.: Simulating landslides for natural disaster prevention. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (1998).

[HW04] HOLMBERG N., WÜNSCHE B. C.: Efficient modeling and rendering of turbulent water over natural terrain. In *Proc. of GRAPHITE '04* (2004), pp. 15–22.

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Proc. of SIGGRAPH '90* (1990).

[KMN88] KELLEY A. D., MALIN M. C., NIELSON G. M.: Terrain simulation using a model of stream erosion. In *Proc. of SIGGRAPH '88* (1988), pp. 263–268.

[Lew87] LEWIS J. P.: Generalized stochastic subdivision. *ACM Trans. Graph. 6*, 3 (1987), 167–190.

[LKS*05] LEFOHN A. E., KNISS J., STRZODKA R., SENGUPTA S., OWENS J. D.: Glift: Generic, efficient, random-access GPU data structures. In *Proc. of SIGGRAPH '05* (2005).

[LM93] LI X., MOSHELL M.: Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *Proc. of SIGGRAPH '93* (1993), pp. 361–368.

[Man83] MANDELBROT B. B.: *The Fractal Geometry of Nature*. W.H. Freeman and Company, 1983.

[MDH07] MEI X., DECAUDIN P., HU B.-G.: Fast hydraulic erosion simulation and visualization on GPU. In *Proc. of Pacific Graphics* (2007), pp. 47–56.

[MFC06] MAES M. M., FUJIMOTO T., CHIBA N.: Efficient animation of water flow on irregular terrains. In *Proc. of GRAPHITE '06* (2006), pp. 107–115.

[MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The synthesis and rendering of eroded fractal terrains. In *Proc. of SIGGRAPH '89* (1989), pp. 41–50.

[Nag97] NAGASHIMA K.: Computer generation of eroded valley and mountain terrains. *The Visual Computer 13*, 9–10 (1997).

[NWD05] NEIDHOLD B., WACKER M., DEUSSEN O.: Interactive physically based fluid and erosion simulation. In *Proc. of Eurographics Workshop on Natural Phenomena* (2005).

[OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Proc. of Computer Animation* (1995).

[ON00] ONOUE K., NISHITA T.: A method for modeling and rendering dunes with wind-ripples. In *Proc. of Pacific Graphics* (2000), pp. 427–428.

[ON03] ONOUE K., NISHITA T.: Virtual sandbox. In *Proc. of Pacific Graphics* (2003), pp. 252–260.

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. In *Proc. of SIGGRAPH '89* (1989), pp. 253–262.

[PH93] PRUSINKIEWICZ P., HAMMEL M.: A fractal model of mountains with rivers. In *Proc. of Graphics Interface* (1993).

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer–Verlag, 1990.

[RPP93] ROUDIER P., PEROCHE B., PERRIN M.: Landscapes synthesis achieved through erosion and deposition process simulation. *Comp. Graph. Forum 12*, 3 (1993).

[ŠBK08] ŠŤAVA O., BENEŠ B., KŘIVÁNEK J.: Interactive hydraulic erosion on the GPU. In *ShaderX^7 – Advanced Rendering Techniques (to appear)* (2008), Charles River Media.

[SFK*07] SELLE A., FEDKIW R., KIM B., LIU Y., ROSSIGNAC J.: An unconditionally stable MacCormack method. *J. Sci. Comput.* (2007).

[SOH99] SUMNER R. W., O'BRIEN J. F., HODGINS J. K.: Animating sand, mud, and snow. *Comp. Graph. Forum 18*, 1 (1999).

[SS05] STACHNIAK S., STUERZLINGER W.: An algorithm for automated fractal terrain deformation. In *Proc. of Computer Graphics and Artificial Intelligence* (2005).

[Ter07] TERRAGEN: Terragen. http://www.planetside.co.uk/terragen/, 2007.

[WCMT07] WOJTAN C., CARLSON M., MUCHA P. J., TURK G.: Animating corrosion and erosion. In *Eurographics Workshop on Natural Phenomena* (2007).

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain synthesis from digital elevation models. *IEEE Trans. Visual. Comp. Graph. 13*, 4 (2007), 834–848.

© The Eurographics Association 2008.