Fast Hydraulic and Thermal Erosion on GPU

Balázs Jákó and Balázs Tóth

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics, Budapest/Hungary

Abstract

Computer games, simulators and many other computer graphics applications display external scenes where a realistic looking terrain is a vital part of the view. In this paper we propose a method that generates realistic virtual terrains by simulation of hydraulic and thermal erosion on a predefined heightfield. The model is designed to be executed interactively on parallel architectures like graphics processors.

1. Introduction

Natural eroded terrains have typical features like valleys, riverbeds, ridges, etc. These are the results of different kinds of erosions caused by water, thermal shocks, and wind. Hydraulic erosion is caused by running water on terrain surface coming from falling rain and water sources. The flowing water dissolves soil and transports it to lower locations where dissolved sediment is deposited. Thermal erosion is caused by temperature changes due to the alternation of hot sun and cold night air. Hard surfaces are cracking up to smaller parts by the time, the decomposed material is moving down to lower areas due to gravity.

There are different approaches to generate virtual terrains with features caused by these phenomena. Topographical methods use structural models to simulate water systems. These emphasize water flow, but mountain slopes are less natural in results. Physics based models like ours simulate erosion factors and their effects on terrain surface.

The computational power of modern GPU hardware makes it possible to execute such simulations interactively, allowing observation and manual intervention during the execution. In our method we focus on hydraulic and thermal erosion, because these have the most impact on the terrain surface.

Physics based erosion methods use some kind of fluid simulation. Chiba et al. [CMF98] were the first to propose simulating valleys and ridges using particle systems. Beneš et al. [BTHB06] proposed a model that uses Navier-Stokes equations on a 3D regular grid that simulates the erosion process. Neidhold et al. [GeN*] used a simplified Newtonian physics model for velocity computation on a 2D grid. All of these models are computationally expensive, and contain data dependency making them difficult to execute on parallel hardware. 2D Navier-Stokes equations were solved efficiently by Harris et al. [Har05] and Wu et al. [WLL04]. Kass et al. [KM90] solved shallow water equations in their model, which was extended by Beneš et al. [Ben07] to simulate erosion in real-time. Mei et al. [MDH07] used this model in their simulation by employing *virtual pipes* introduced by O'brien et al. [OH95] that is the key to parallel execution. Papers [SBBK08] and [SBK08] describe a complex model to simulate hydraulic erosion and terrain slippery simultaneously. Our model is partly based on [MDH07].

Similarly to [SBK08], we combine hydraulic and thermal erosion based on [MDH07] and [BF01], applying the pipe idea to the latter. This improves the original hyrdaulic erosion model making the sides of the river steeper. We introduced some additional features that make the model more flexible and accurate even on a single height map layer, including local hardness of the material. Based upon this, we simulate sediment softening. Our parallel version of the thermal erosion model use 8 virtual pipes to make the results more precise.

2. Erosion Model

Our hydraulic erosion model is an improved version of the method introduced by Mei, Decaudin and Hu [MDH07]. This model runs on a 2D uniform Euclidean grid and uses the following quantities in each (x,y) cell (see Figure 1):

- terrain height b and water height d,
- suspended sediment amount s,
- water outflow flux $\mathbf{f} = (f^L, f^R, f^T, f^B)$,

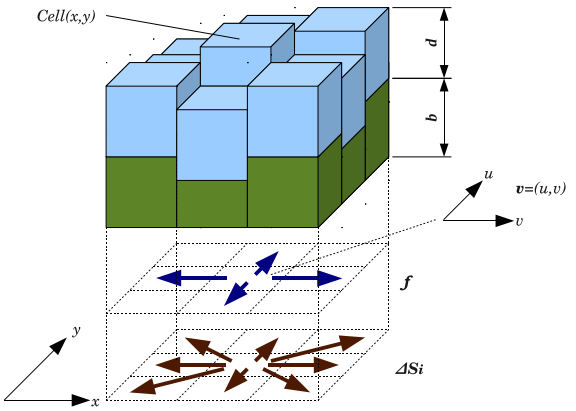


Figure 1: *Water and thermal sediment flow model.*

- velocity vector $\overrightarrow{\mathbf{v}}$,
- thermal erosion outflow flux $\mathbf{S} = (s^i), i = \{1..8\}$.

These values are updated in each iteration. The simulation iteration consists of the following steps:

1. Water incrementation due to rain or water sources.
2. Flow simulation using shallow-water model. Computation of velocity field and water height changes.
3. **Soil flow calculation with outflow in virtual pipes of the thermal erosion model**.
4. Computation of erosion-deposition process with velocity field.
5. Transportation of suspended sediment by the velocity field.
6. **Thermal erosion material amount calculation**.
7. Water evaporation.

The steps of the thermal erosion are in bold. These all are executed in each iteration, gradually changing the state variables in each cell [SKS01]. Let $b_t, d_t, s_t, f_t, \overrightarrow{v_t}$ denote the data at a given time t and Δt the time increment in the current iteration. In the following, we summarize the calculations producing the values in the next $t+\Delta t$ time. Since the model calculates some variable values in two or more steps, we will use subscripts $1, 2, \ldots$ to distinguish the temporal values from the final output used in the following iterations.

First, we simulate the effects of water arriving to the terrain surface. Unlike the original model, we use constant $r(x,y)$ rain rate for each cell instead of large randomly distributed raindrops. This specifies the water amount arriving at a given (x,y) cell during a Δt time. This gives us more balanced and finer grained results on the long run. The water height is updated by the following formula:

$$d_1(x,y) = d_t(x,y) + \Delta t \cdot r_t(x,y) \cdot K_r \tag{1}$$

where K_r is a global simulation parameter that scales the overall rate of water increment and d_1 is the intermediate value of the water height.

After that, we calculate the water flow between cells. Each (x,y) cell has four virtual pipes to the four neighbors which transport water outward from the given cell. Water can not flow backwards. The neighboring cells have four virtual pipes also transporting water to opposite directions. The water outflow flux is updated with the pressure difference between interconnected cells. Let $\mathbf{f} = (f^L, f^R, f^T, f^B)$ denote the outflow flux in a given (x,y) cell, where f^L is the outflow flux to the left neighbor at $(x-1,y)$, and similarly f^R, f^T, f^B are the outflow fluxes to right, top and bottom directions, respectively. We calculate the change of f^L:

$$f^L_{t+\Delta t} = \max\left(0, f^L_t(x,y) + \Delta t \cdot A \frac{g \cdot \Delta h^L(x,y)}{l}\right) \tag{2}$$

where A is the cross section area of the virtual pipe, g is the gravity acceleration, l is the length of the virtual pipe and $\Delta h^L(x,y)$ is the height difference between the left and the current cell. Calculation of f^R, f^T, f^B is done in a similar way. The total outflow is maximized by the total amount of the water in the given cell. If the calculated value is larger than the current amount in the given cell, then $\mathbf{f}$ will be scaled down with an appropriate factor K:

$$K = \max\left(1, \frac{d_1 \cdot l_x \cdot l_y}{(f^L + f^R + f^T + f^B) \cdot \Delta t}\right) \tag{3}$$

where l_x, l_y are the distances between the grid cells in the x, y directions. The outflow flux is multiplied by K:

$$f^i_{t+\Delta t}(x,y) = K \cdot f^i_{t+\Delta t},\ i = L,R,T,B \tag{4}$$

We calculate ΔV water height change with summarizing f_{out} output and f_{in} input flow values in each (x,y) cell:

$$\begin{aligned} \Delta V(x,y) &= \Delta t \cdot \left(\sum f_{in} - \sum f_{out}\right) = \\ &= \Delta t \cdot (f^R_{t+\Delta t}(x-1,y) + f^T_{t+\Delta t}(x,y-1) + \\ &\quad f^L_{t+\Delta t}(x+1,y) + f^B_{t+\Delta t}(x,y+1) - \\ &\quad \sum_{i=L,T,R,B} f^i_{t+\Delta t}(x,y)) \end{aligned} \tag{5}$$

Then, we update the water height in the current (x,y) cell:

$$d_2 = d_2(x,y) + \frac{\Delta V(x,y)}{l_x l_y}. \tag{6}$$

Using outflow flux values, we can calculate the $\overrightarrow{\mathbf{v}}$ velocity field that needed to calculate hydraulic erosion and deposition. The calculation of the x component is:

$$\Delta W_x = \frac{1}{2}(f^R(x-1,y) - f^L(x,y) + f^R(x,y) - f^L(x+1,y)) \tag{7}$$

The y component is calculated in a similar way.

Now, we can calculate C water sediment transport capacity that represents how much sediment can be transported in a cell with 3D collision between water and terrain surface:

$$C(x,y) = K_c \cdot (-\overrightarrow{\mathbf{N}}(x,y) \cdot \overrightarrow{\mathbf{V}}) \cdot |\overrightarrow{\mathbf{v}}(x,y)| \cdot l_{max}(d_1(x,y)) \tag{8}$$

where $\vec{\mathbf{N}}(x,y)$ is the terrain surface normal at point (x,y) and $\vec{\mathbf{V}}$ is the 3D water flow vector calculated from surface tangent and $\vec{\mathbf{v}}$ 2D velocity vector.

Deep water floors is practically never eroded by moving water, even if there is a stream, because water flow is slowed down by depth. To simulate this, we added an l_{max} limiting function to this fomula. This function continuously scales down the fluid erosion effects by the water depth. This produces more natural looking sea floors than the original model in deeper water.

At this point, the model makes a decision using C capacity. If the s_t transported sediment in cell (x,y) is smaller than C, then we dissolve some soil in the water:

$$b_{t+\Delta t} = b_t - \Delta t \cdot R_t(x,y) \cdot K_s(C - s_t) \tag{9a}$$
$$s_1 = s_t + \Delta t \cdot R_t(x,y) \cdot K_s(C - s_t) \tag{9b}$$
$$d_3 = d_2 + \Delta t \cdot R_t(x,y) \cdot K_s(C - s_t) \tag{9c}$$

where K_s is the global coefficient. Otherwise, if $C < s_t$, then we dispose some of the transported sediment in a similar way:

$$b_{t+\Delta t} = b_t + \Delta t \cdot K_d(s_t - C) \tag{10a}$$
$$s_1 = s_t - \Delta t \cdot K_d(s_t - C) \tag{10b}$$
$$d_3 = d_2 - \Delta t \cdot K_d(s_t - C) \tag{10c}$$

where K_d is a global parameter controlling deposition speed. Equations 9c and 10c did not exist in the original model, we added these to improve stability. Without these, the erosion deposition step has some unwanted feedback to water flow simulation and causes some regular ripples on the water surface on the long run.

In nature, moving sediment becomes softer by the time. To imitate this, we added a supplemental step to the model, that lowers the $R(x,y)$ local hardness coefficient of the terrain when some soil disposed:

$$R_{t+\Delta t}(x,y) = \max(R_{min}, R_t(x,y) - \Delta t \cdot K_h K_s(s_t - C)) \tag{11}$$

where R_{min} is the lower limit of hardness, K_h is a global coefficient controlling the sediment softening. The next step in the model is to move dissolved sediment along the water using $\vec{\mathbf{v}} = (v_x, v_y)$:

$$s_{t+\Delta t}(x,y) = s_1(x - v_x \cdot \Delta t, y - v_y \cdot \Delta t) \tag{12}$$

If $(x - u \cdot \Delta t, y - v \cdot \Delta t)$ is not on the grid, the model uses linear interpolation amongst the four closest grid points. In the last step, we simulate water evaporation:

$$d_{t+\Delta t}(x,y) = d_3(x,y) \cdot (1 - K_e \Delta t) \tag{13}$$

Our thermal erosion model is similar to the hydraulic erosion, but it uses 8 pipes. Let's denote the terrain height of the current (x,y) cell by b and its eight neighbors by $b^i, i = 1,2,...,8$, and the height difference between the current cell and its lowest neighbor by $H = \max\{b - b^i, i = 1,...,8\}$. The area of the cells is a and the volume to be moved is $\Delta S = a \cdot H/2$. This is the maximum otherwise the algorithm will oscillate. To handle local $R(x,y)$ terrain hardness, we have extended this formula:

$$\Delta S_{t+\Delta t} = a \cdot \Delta t \cdot K_t \cdot R_t(x,y) \cdot H/2 \tag{14}$$

where K_t is a global coefficient. Then we move this amount to the lower neighbors proportionally if the so called talus angle is larger than that the value determined by material viscosity. Let's denote the distance between two cells by d, the talus angle by $\alpha = \arctan(b - b^i)/d$ and the set of neighbors that are lying lower than the current element under the talus angle by $A = \{b^i, b - b^i < 0 \wedge (b - b^i)/d > \arctan\alpha, i = 1,...8\}$. Each element in A will get part of the volume ΔS_i proportional to its height difference:

$$\Delta S_i = \Delta S \frac{b^i}{\sum_{\forall b^k \in A} b^k} \tag{15}$$

Similarly to [SBK08], we do not move ΔS_i volumes directly to cells in set A, we use the pipe model instead to make the model parallel (see Figure 1). Separate simulation steps update the terrain height for each cell by summarizing the incoming material flow from their neighbors via the pipes.

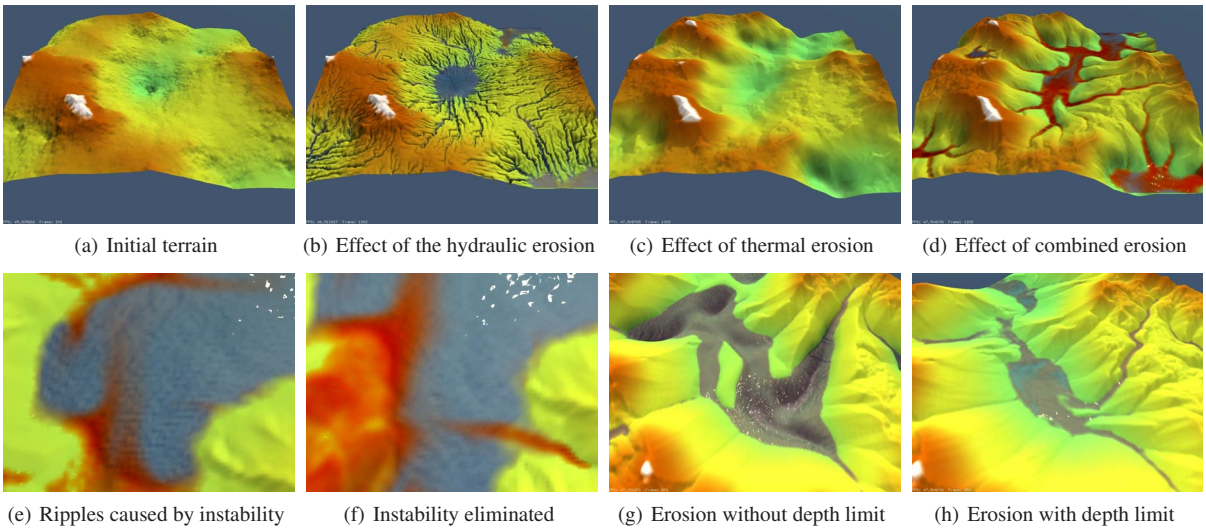
3. GPU Implementation

Cell structure is represented by 2D 4-channel floating-point texture layers stacked upon each other, attached to a single framebuffer object. The texels at the same position in the texture layers are consisting one cell, containing all the simulation variables associated with it. With two framebuffers we calculate one iteration using one of these buffers as an input and the other as an output [SKSS08]. After the iteration we swap these buffers and start over. The iteration process is implemented in a single fragment shader that runs in multiple passes on a full-size quad over the entire framebuffer, writing the output textures as multiple render target. The model utilizes linear interpolation and edge wrapping. The texture buffers can be used directly at the terrain rendering using terrain height values to offset and coloring of triangle vertices of rendered terrain and water mesh.

4. Results and conclusions

Figure 2 demonstrates the effects of our combined erosion process on a random terrain. The initial terrain was generated by the iterative version of the well-known Diamond-square algorithm. The hydraulic erosion carves deep grooves into the terrain surface. With this extended model we can achieve more realistic results. Our method runs with 65 iterations per second on an 1024×1024 terrain on an ATI HD4870 graphics card.

We proposed an improved erosion model that can be executed on the GPU. The method combines and improves two algorithms to simulate hydraulic and thermal erosion. The original hydraulic erosion method is extended to be more



(a) Initial terrain (b) Effect of the hydraulic erosion (c) Effect of thermal erosion (d) Effect of combined erosion

(e) Ripples caused by instability (f) Instability eliminated (g) Erosion without depth limit (h) Erosion with depth limit

Figure 2: *Effects of the different components of our method on random terrain after 1000 iterations. The reddish color indicates sediment in the water. Slopes in valleys on eroded terrains are less steep than at higner areas due to material softening.*

versatile and stable. The thermal erosion model is a enhanced version of an earlier work to make it able to produce more realistic results and was integrated with the fluid erosion model.

Acknowledgements

This work has been supported by OTKA K-719922 and by TÁMOP-4.2.1/B-09/1/KMR-2010-0002.

References

[Ben07] BENEŠ B.: Real-time erosion using shallowwater simulation. *The 4th Workshop on Virtual Reality Interactions and Physical Simulation - Vriphys'07* (2007), 43–50. 1

[BF01] BENES B., FORSBACH R.: Layered data representation for visual simulation of terrain erosion. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics* (Washington, DC, USA, 2001), IEEE Computer Society, p. 80. 1

[BTHB06] BENEŠ B., TĚŠÍNSKÝ V., HORNYŠ J., BHATIA S. K.: Hydraulic erosion: Research articles. *Comput. Animat. Virtual Worlds 17*, 2 (2006), 99–108. 1

[CMF98] CHIBA N., MURAOKA K., FUJITA K.: An erosion model based on velocity fields for the visual simulation of mountain scenery. *Journal of Visualization and Computer Animation 9* (1998), 185–194. 1

[GeN*] GALIN E., (EDITORS P. P., NEIDHOLD B., WACKER M., DEUSSEN O.: Interactive physically based fluid and erosion simulation. 1

[Har05] HARRIS M.: Fast fluid dynamics simulation on the gpu. In *ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), SIGGRAPH '05, ACM. 1

[KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, September 1990), vol. 24 of *SIGGRAPH '90*, ACM, pp. 49–57. 1

[MDH07] MEI X., DECAUDIN P., HU B.-G.: Fast hydraulic erosion simulation and visualization on GPU. In *15th Pacific Conference on Computer Graphics and Applications, Pacific Graphics 2007, November, 2007* (Maui, Hawaii, Etats-Unis, Nov. 2007), IEEE, pp. 47–56. 1

[OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Proceedings of the Computer Animation* (Washington, DC, USA, 1995), IEEE Computer Society, pp. 198–. 1

[SBBK08] STAVA O., BENES B., BRISBIN M., KRIVANEK J.: Interactive terrain modeling using hydraulic erosion. Gross M., James D., (Eds.), Eurographics Association, pp. 201–210. 1

[SBK08] STAVA O., BENES B., KRIVANEK J.: Interactive hydraulic erosion on the gpu. In *ShaderX 7* (2008), Engel W., (Ed.), East River Media. 1, 3

[SKS01] SZIRMAY-KALOS L., SZÉCSI L.: *General Purpose Computing on Graphics Processing Units*. MondAT kiado, 2001. 2

[SKSS08] SZIRMAY-KALOS L., SZÉCSI L., SBERT M.: *GPU-Based Techniques for Global illumination effects*. Morgan and Claypool, 2008. 3

[WLL04] WU E., LIU Y., LIU X.: An improved study of real-time fluid simulation on gpu. *Department of Computer Science, University of Manchester, UK. Since 15* (2004), 139–146. 1