Hydraulic Erosion of Soil and Terrain

Atira Odhner, Joseph Fleming

2009

Abstract

We have developed an algorithm for hard and soft water-based terrain erosion. Additionally, we have added a rain generation into the simulator. The algorithm is based upon fluid simulation, but is simplified, while still preserving the fluid motion effects.

Introduction

Our goal is to design an algorithm that generates water-eroded terrain quickly for the purposes of artistic productions, rather than scientific study. We wish to simulate the effects of rivers and streams, taking into account erosion, transportation, and sedimentation. Other algorithms take into account full fluid simulation, but we wished to obtain similar results by using a less-costly yet water-based erosion algorithm.

Previous Work

There are many examples of previous work on the subject of erosion, as it is important for both the graphics community and the environmental conservation community. We have chosen a few that are both relevant to the our algorithm, and demonstrate natural and atmospheric effects related to our algorithm. Since there are so many similar works (often repeating themselves), these studies are introduced by relevance, rather than by year.

In 2005, Beneš and Arriaga [1] proposed a method for generating table mountains. Table mountains are desert geological structures that are very narrow, yet water-based erosion; plateaus with almost vertical cliff walls. These mountains form through mostly thermal erosion rather than by; rock that is exposed deteriorates. For this reason, table mountains are found in desert environments. The algorithm in this paper is not water-based, but it relates to the erosion of river banks. The type of erosion introduced is *diffusion erosion*. This is a gradual "settling" of deposited material (sediment) over time. This happens on river banks as more and more of the material is taken away. This isn't exactly the settling of sediment, but oftentimes a riverbank is made of soil or sand, which is a form of sediment in itself.

In 2006, Valette et al. [3] made a soil erosion model dealing with rainfall. The model was tailored to scientific studies. For our model, we decided to do a "simulated" rainfall generator to see what kind of results would be generated.

In 1998, Chiba et al. [2]. developed a water-based erosion algorithm that used velocity fields. The velocity field feature is useful because it allows for flow patterns, like in full fluid simulations. The simulation consists of a regular 2D height grid, an array of values related to the activity of the fluid, each relating to a "cell" of the height grid. The velocity field is determined by the height of the underlying terrain. Particles are used not as markers, but as the actual water particles contributing to the simulation. As the water particles collide with the underlying terrain, they pick up material (erosion). Deposition (or *sedimentation*) is determined per-cell of the height grid, averaged from all the particles moving in it (or through it). It is this model that we based our algorithm off of.

Basic Algorithm

Our algorithm is a modification of Chiba et al. The basic algorithm described by Chiba et al. has

one 2D height map across a regular grid. A number of water particles are placed at grid points. Each water particle moves in the direction of the greatest incline. Acceleration from gravity is also added into the velocity. The acceleration is determined by:

$$|\mathbf{a}| = \frac{\mathrm{d}h}{\sqrt{len^2 + \mathrm{d}h^2}} G,$$

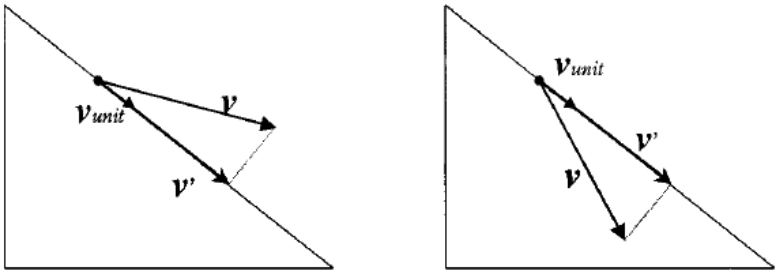
where dh is the difference in height between the current position and the steepest neighbor grid point, *len* is the distance to that grid point, and G is acceleration due to gravity. When the position of the particle moves through the plane of the grid cell, a collision has occurred. This does not require any sort of ray tracing. Rather, the new height of the particle is compared with the interpolated height at that point. The energy from the collision is determined by:

$$e = m\Delta v^2 / 2,$$

where m is the mass of the particle (in the case of water, it is 1.0), and Δv is the decrease in velocity from the previous step. When the velocity is not tangent to the incline of the grid cell, a simple operation called projection is performed on the velocity vector (this is also known as obtaining the "sliding" vector) to conform the direction to that of the incline, while keeping the appropriate velocity magnitude:

$$v' = (v_{unit} \cdot v) v_{unit},$$

where v_{unit} is the normalized velocity vector of v. This is illustrated by the following diagram from the original paper by Chiba et al.:



The collision energy, number of water particles, and overall velocity from all particles in each cell is summed up and stored in 2D arrays corresponding to each type of value (energy, water, and velocity). The velocity is averaged over all the particles in the cell. These values are later used to determine the flow, amount of carried sediment, and amount of deposited sediment for that cell. Our model keeps track of all these values, but they tend to be processed and stored per-particle.

The original model does erosion, transportation, and sedimentation on a per-cell basis. Our model does it on a per-particle basis. This increases performance when there are fewer water particles than terrain cells, and more importantly, increases the versatility of our system; our system does a much better job of modeling streams and rivers. The next stage of the algorithm—transportation and sedimentation—uses the same principles as the original, but adapted to our per-particle model.

Each particle stores its 3D velocity to use for collision detection and stores a 2D position and for every iteration the following steps occur for each particle:

1. The particle is deleted if it is outside the boundaries of the height field, or as determined by the rate at which particles are removed. This rate is determined by the

input file. It prevents particles that are stuck to disappear rather than remain stuck forever.

2. The particle decides how much material to drop.

This amount is added to the height of the soft material. All deposited material becomes soft material because more compact rocks would not form immediately from sedimentation.

3. The height of the water is stored

4. The Force down the plane is calculated and added to the particle's velocity:
   $mg\ G_n - mg(G_n) \cdot N$
   where mg is the weight of the water particle (mass*gravity)
   G_n is the normalized vector corresponding to gravity
   N is the normal of the plane where the particle is positioned

5. The particle's velocity is added to its position

6. A pointer to the particle is added to the corresponding grid square

7. A check is performed to determine if the particle collided with the ground:

It collides if the difference between the height of the ground at its new position and the height it would be at if it had proceeded using its 3D velocity (use the stored height of the water plus the vertical component of the particle's velocity) is greater than the amount of water stored in the grid cell so far. (Amount of water = number of water particles * size of a water particle.) This check makes it so that cells with large amounts of water do not treat each water particle as colliding with the cell, which overcomes some of the shortcomings of using a height field without expense.

8. If a collision is detected the velocity of the particle is reflected off the normal of the cell and decreased by some percentage. This lost energy is the energy of collision, which is then added to the cell. The water particle also picks up some mass which is removed from the height of the grid.

Once each particle has been dealt with, each cell of the grid is storing the necessary information to perform the second half of the erosion process. It is storing pointers to all the particles it contains and it is storing the total energy of collisions during this time step. A couple of simple steps are applied to each cell:

1. For each cell in the grid, the amount of material to be removed is:
   Ke*Total Energy of Collisions in the Cell / (Number of Water Particles in the Cell)^2

2. The material removed from the cell is then divvied up between the water particles in the cell.

Our model also uses multiple height map layers. One is for the original topology, and the second is for deposited material. We will refer to these layers as the *hard* and *soft* layer. The model is set up to allows colliding particles to take material from either layer, providing continuous erosion. Separate K_e and K_q properties for these layers allow for different behavior when each layer is eroded. Deposited and

transported sediment is always treated as *soft* though, and is always added to the *soft* layer.

Note: height values of water particles are calculated by interpolating the heights of the corners of the grid square it is in. The heights of the corners are calculated by comparing the soft and hard layer height values at that position and taking the larger value.

We also implemented rainfall as described by Gilles et. al. Particles are generated in a circle determined by the user, as with all our water generation, but the particles then pick up material and deposit it within a small radius determined by their reflectance off the normal where they hit.

Our algorithms is O(cpt) in terms of the number of cells in the grid, number of particles generated per time step, and the number of time steps.

Results

The algorithm runs fairly quickly. Most of the time is spent in loading the environment. In Figure 1, you can see the basic test case. Water pours in from the top, erodes, and exits out the bottom. Deposition can be seen in the lower left.

In Figure 2, an effect called river meanders is demonstrated. River meanders is a natural effect that occurs when water runs down a gently sloping trough. The water swerves back and forth, like a sine wave, across the lowest part, carving a deep trench.

Figure 5 illustrates the rain test case. Rain erodes the soft material on top first, making jagged points, and then smooths out when the hard layer is reached below.

During the development of our program, we encountered several problems. The basic algorithm didn't account for evaporation, and this caused many particles to get "stuck" and dig deep holes without ever stopping. To resolve this a lifespan was added to the particles.

Conclusion

Overall this algorithm does a very good job at creating erosion patterns. The deposition stage could use some refining in the future. The model used here may need further tuning to make it completely particle based.

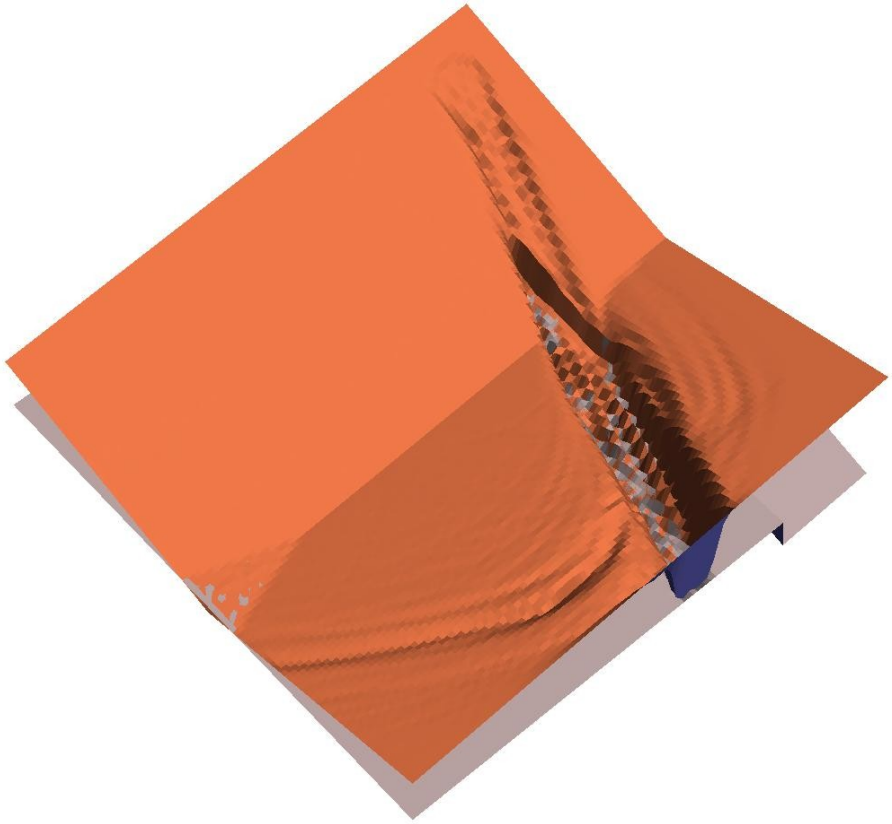


Figure 1: Basic Test case. Sedimentation can be seen at the lower left.

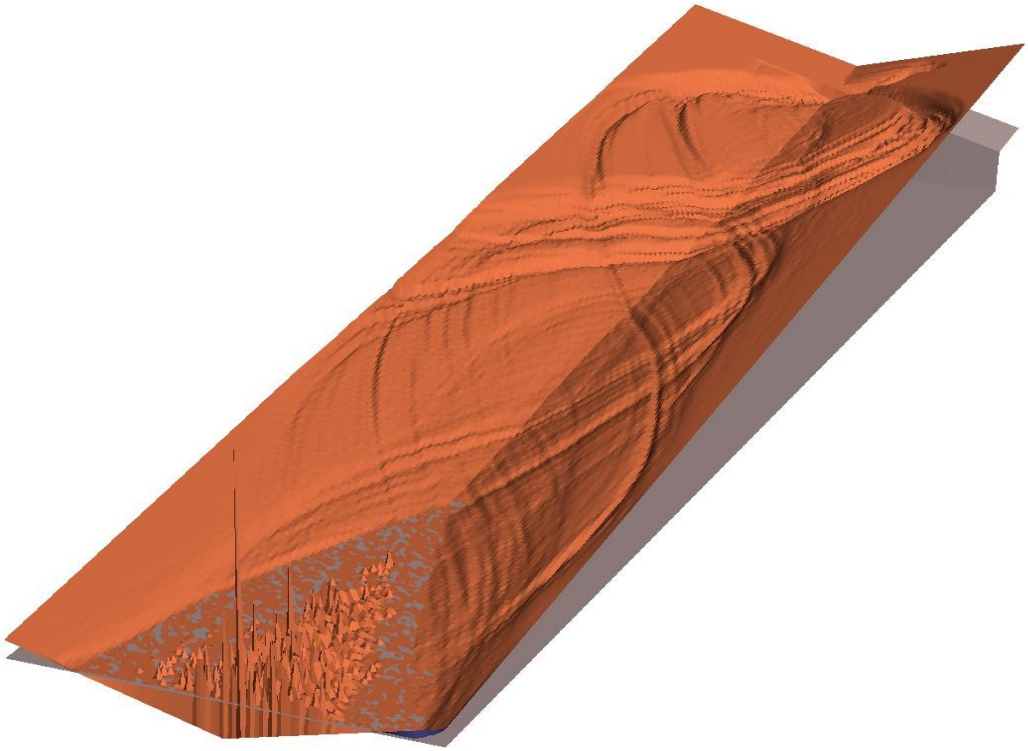


Figure 2: River Meanders. Deposition bug can be seen at the lower left.

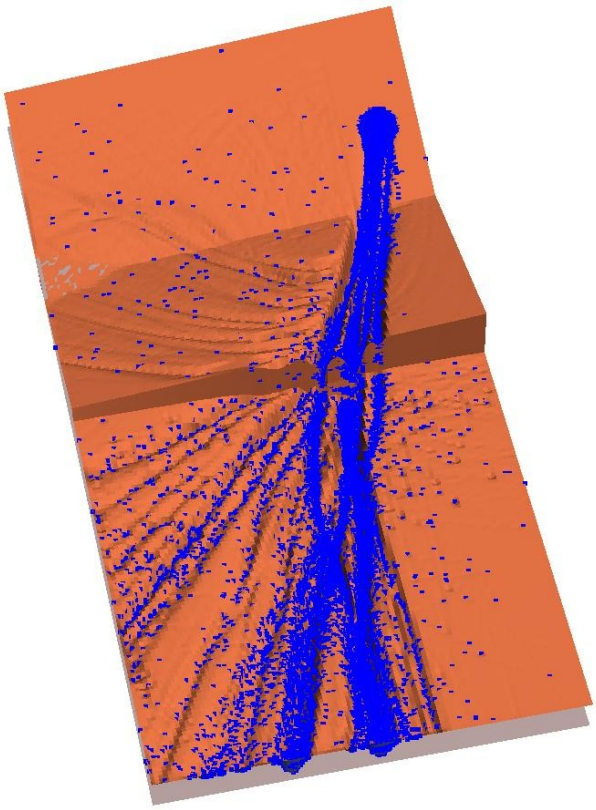


Figure 3: A river emptying into a basin.

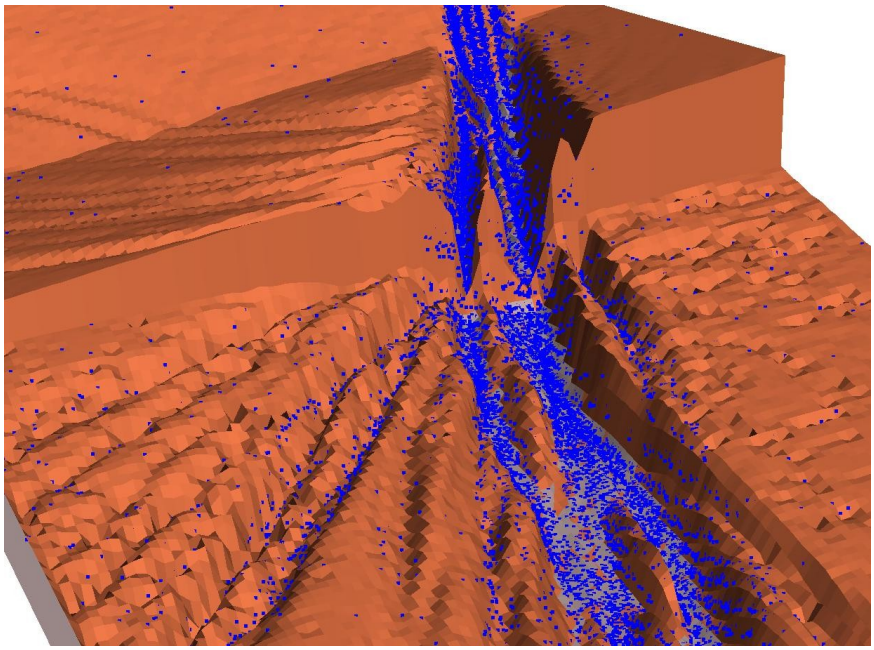


Figure 4: Closeup of Figure 4. "Waterfall" effect with outcroppings.

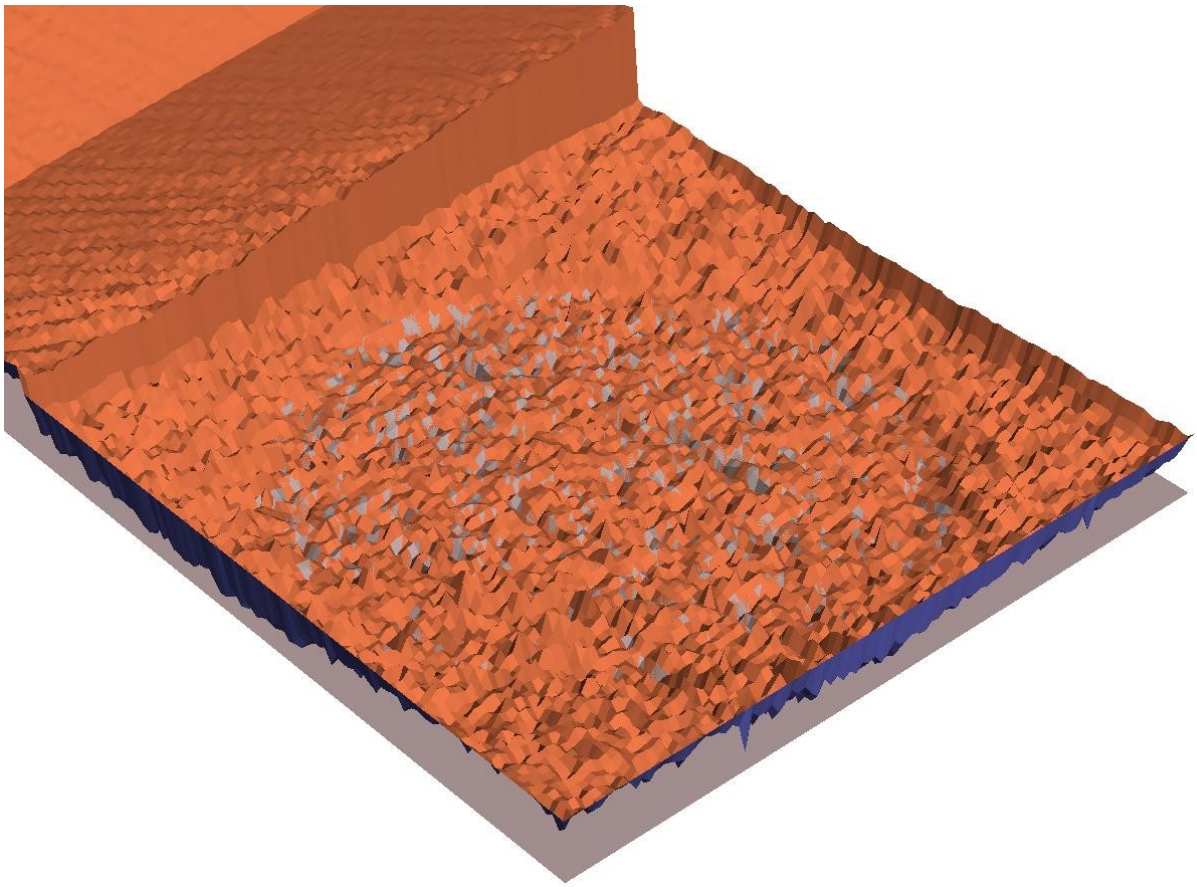


Figure 5: Rain generation. The soft top layer erodes away in a spiky pattern until the hard layer is reached underneath, smoothing it out.

Bibliography

1. N. Chiba, K. Muraoka, and K. Fujita. "An Erosion Model Based on Velocity Fields for the Visual Simulation of Mountain Scenery." THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION 9(1998): 185-194.
2. B. Beneš, and X. Arriaga. "Table Mountains by Virtual Erosion." Eurographics Workshop on Natural Phenomena (2005): 33-39.
3. Gilles Valette, Stéphanie Prévost, Laurent Lucas, and Joël Léonard. "SoDA project: A simulation of soil surface degradation by rainfall." Computers & Graphics. 30(2006): 494-506.