Interactive simulation of hydraulic erosion on the GPU

Juraj Vanek, Ondrej Stava, Bedrich Benes
Department of Computer Graphics Technology, Purdue University, 401 N. Grant St., West Lafayette, IN 47907
vanek@purdue.edu, stava@adobe.com, bbenes@purdue.edu

High Performance Computer Graphics Laboratory

PURDUE UNIVERSITY

NVIDIA CUDA™ TEACHING CENTER

Abstract
Terrain modeling is an important task in digital content creation and physics-based approaches have the potential to simplify it significantly. We introduce a new interactive, intuitive, and accessible physics-based framework for digital terrain editing. A terrain, composed of layers of materials, is edited with interactive modeling tools built upon different physics-based erosion and deposition algorithms. The user has a great level of control over the process and receives immediate feedback since the GPU-based erosion simulation runs in real-time and is fully interactive.

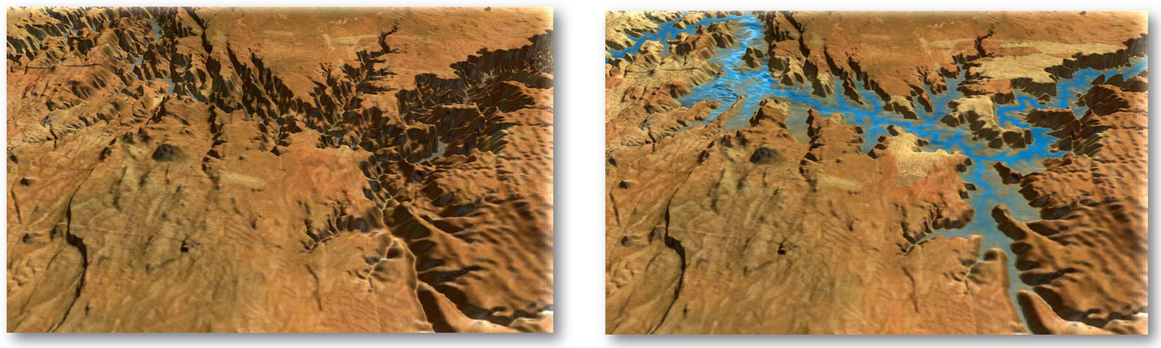

Figure 1: Digital elevation model of *Grand Canyon* interactively flooded and eroded by our physics based tools

1. Motivation
Creating a realistic looking terrain is not an easy task. Although one can use digital elevation models of existing terrains (DEM), in many cases they need to be further edited. Editing of such models manually can be tedious task. Simulations like hydraulic erosion have potential to make editing easier but problem with such methods is their computational intensity

We introduce an easy-to-use framework for editing of large terrain using intuitive physics-based tools like local rain, creating water outlets, erosion and evaporation. System is interactive and changes are immediately visible. Input data are also variable, they could have a form of existing height maps or be procedurally generated.

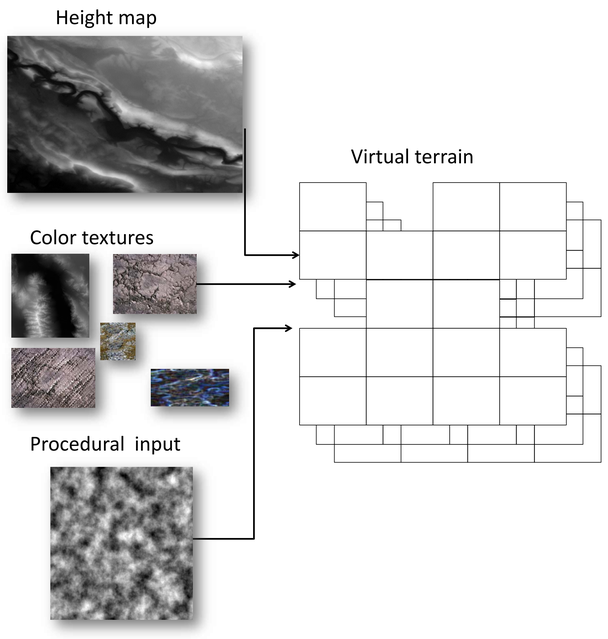

Figure 2: Pre-processing stage

2. System inputs
Our system supports various inputs like digital elevation maps, color textures or virtually any terrain achieved by procedural generator. Despite of data origin, all these inputs can be **combined** together (addition or subtraction)

We can have **multiple layers of material** (e.g. rock, soil, mud, as seen on Figure 3). Each layer has different properties affecting the water simulation – total thickness, dissolving and deposition rate and of course color or texture for rendering.

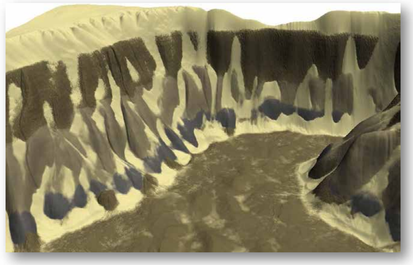

Figure 3: Multiple layers of material

3. Pre-processing
Terrain is divided into **tiles** for adaptive computation. Resolution of tiles is based on the terrain differences within the tile. Big differences (like mountains) result in high data resolution, low differences (flat plain) result in low resolution.

After the tiling, we call this structure **virtual terrain**. Data for each tile are uploaded into the **GPU** memory as data textures (Figure 2).

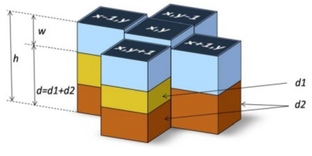

Figure 4: Virtual pipe model. Each cell has information about terrain layers, actual water height and water flow from neighboring cells

4. Simulation
Runs entirely on the GPU using **fragment shaders** as compute units and data textures. Each tile is divided into grid with small cells (**virtual pipe model**, Figure 4) containing all information about terrain and water levels. Each simulation step updates those cells and after that, analysis is made which cells should be update in the next step (Figure 5). We can consider this as a discretization of **Navier-Stokes equations** for fluid simulation.

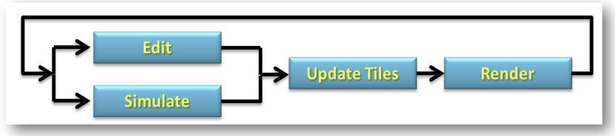

Figure 5: After each step, system responds to edits made by user and decides which tiles should be updated in the next simulation step (e.g. change resolution)

5. Physic based tools
User can use various tools to **shape the terrain**. Each layer can be selected and pulled up/down to change the thickness. Anywhere on the terrain, the new water outlet can be created. If only a small amount of water input is needed, there is an option to turn on rain over small area. When the user is satisfied with editing result, water can be evaporated and resulting terrain exported.

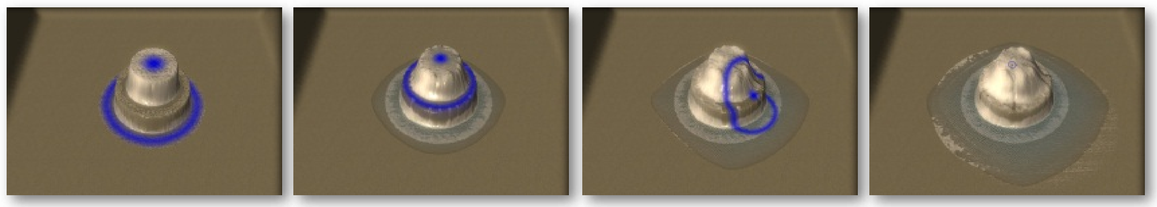

Figure 6: Example of shaping terrain using just a few tools (pulling, rain and evaporation)

6. Rendering
We use **OpenGL 4** to render the results of simulation by displacing tiles based on terrain and water level, blending color of all layers together and various effects like water reflection and refraction and HDR rendering.

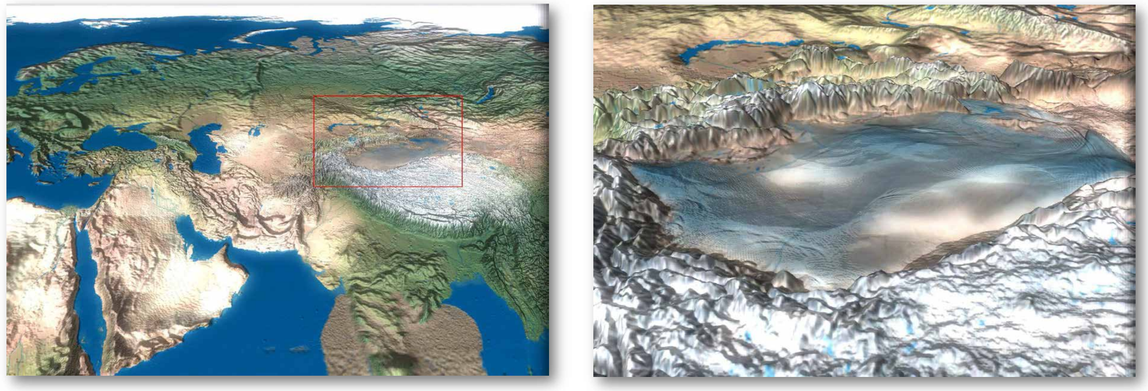

Figure 7: Example of editing very large terrain (12k x 12k image data size, three layers of material). Even with this size it is possible to edit it interactively

7. Results
We have tested our framework on various input data. Thanks to an adaptive tiling system, our framework was able to handle **very large terrain data** (Figure 7) at still interactive frame rates. For example, average simulation time per one simulation step at flooded *Grand Canyon* model in Figure 1 was **22.7 milliseconds** (which is almost 45 frames per second) on **GPU *GeForce GTX480*** and the whole simulation took less than three minutes (with all editing operations).

ACKNOWLEDGMENTS
We would like to thank NVIDIA for providing graphics hardware. This work has been supported by NSF IIS-0964302, NSF OCI-0753116 Integrating Behavioral, Geometrical and Graphical Modeling to Simulate and Visualize Urban Areas and by the research program LC-06008 (Center for Computer Graphics)