REAL-TIME WATER IN A LANDSCAPE

Adam Vlček
Doctoral Degree Programme (1), FIT BUT
E-mail: xvlcek12@stud.fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

ABSTRACT

The complexity and dynamism of virtual reality is continuously improving. This work examines the impact of real-time water simulation in large landscapes. The main goal was to find a quick well looking approximations rather than a complex physically correct model, so a combination of various techniques is being used. The main topic is water movement in the terrain and its effects on it like erosion, snow melting and moisture impact on vegetation. The usage of modern programmable GPU not only for visualization but also as a powerful simulation instrument is also covered.

1 INTRODUCTION

As the computation power of personal computers increases, the virtual reality becomes more and more truly realistic. Water is one of the most important elements of the nature, and also one of the hardest to simulate. Nevertheless, it is everywhere: clouds, fog, rain, snow, streams, rivers, lakes, seas and oceans. Moreover, water is also under the surface. The amount of water in the soil significantly affects the vegetation and soil consistency. Water is also one of the most important terrain forming forces.

In today virtual reality, water and all the associated phenomena are usually static, which greatly simplifies the problem. There is usually little or no simulation and visualization of static data is also much simpler. Because the water state in nature is in most cases changing slowly, this approach is usually good enough. However, water can be also very dynamic, for example when a dam breaks or tsunami hits the shores. The recent advances in real-time physics simulation made it possible to use particle based fluids in limited areas [1]. This is mainly due to the shift in GPU hardware to the general purpose computing – GPGPU. Some work has been also done on the larger scales [2]. Another approach to physical simulation of fluids is cellular automaton [3], which also maps extremely well to the GPU.

Particles in real-time simulations are far from particles in physics. They behave more like big balls and their amount is many orders lower. However, they can produce very nice effects, especially when the water distribution is not uniform in the space. While most particle systems are computed in 3D and with sophisticated collision detection, it is also possible to use non-colliding particles sliding on the 2D height field [4]. Once trapped in a depression, they can

form a lake object, which will consume all arriving particles to increase it's level, eventually transporting the particles to the outflow.

Cellular automatons divide the space into a regular grid of discreet cells, representing the average state of the environment over the area or volume covered by the cell. The next state of a cell is computed from the cell's state and the state of a defined neighborhood. Cellular automatons are more suitable for simulation on the whole terrain rather than just the areas with rivers and lakes. While too complex in full 3D, in 2D they can cover larger area than particle systems. They are also very simple and map extremely well on the GPU hardware.

This article is based on my bachelor [5] and diploma [6] theses. A very similar work [7] has also been published recently, so the differences will be discussed.

2 USED TECHNIQUES

The goal was to simulate rain on the whole terrain, which should flow down the hills, forming streams, rivers and lakes, while performing water erosion on the whole terrain. Because the water and erosion should be computed on the whole terrain, the cellular automaton was chosen. Since terrain is usually defined by a height field in real-time applications, the 2D cellular automaton cells can be easily mapped on the points of the height field.

For the purpose of water simulation, we need the following information for every cell: terrain height, water column, water speed in N directions. The number N is given by the size of the neighborhood. The pipe model is being used [7], which creates virtual pipes connecting neighboring cells and computes the flow between them in two phases:

1. Compute the water surface height difference and the associated pressure. Then alter the pipe speeds.
2. Speeds are also affected by the speed of the incoming water. The resulting speeds are mix of the current state and the incoming water.

Since it is not exact physical model, setting the parameters to look the most natural and preventing the system from crazy behavior can be sometimes tricky, but when set well, the results are good.

2.1 EROSION

When the water is being simulated by a cellular automaton, it is relatively simple to add erosion simulation in a similar way. We may consider adding a few parameters to the cell data, like average water column over some time and the amount of dissolved sediment in the water.

There are several aspects of the water erosion, we should attempt to model. First, the terrain is being dissolved by the running water. The material is then transported with the water and later deposited. Also the watersides are being eroded from the side and the beds of lakes and rivers are being smoothed.

There are several possibilities what we can do. A very good erosion model is presented in [7]. Here just a simple approximation is shown to also produce believable results, although not physically correct. The basic idea is to remove terrain in the places with quickly running

water while smoothing with weights based on the water columns is applied in the areas with still water. The results are illustrated on Figure 1.

2.2 TEXTURING AND VEGETATION

A very important aspect of this work is the dynamic adaptation of the terrain texturing and vegetation cover. Since the whole terrain is dynamic, it is not possible to use just predefined textures and have nice results. The solution is to create a few texturing rules in the terrain fragment shader. We can use parameters like altitude, slope inclination, average water column and sun position. First, we compute coefficients indicating likelihood of each terrain type. For example, rock will be in high altitudes and on rapid slopes. Then, these coefficients are merged into the final result according to the terrain type priority. For example, snow will cover rock in even higher altitudes, but will be melted by water.

Similar approach can be applied on basic vegetation. We can feed the GPU with 2D points, describing where we want some plants. The geometry shader computes the height from current terrain height field and evaluates the same set of rules as the terrain texturing fragment shader. When the terrain type is suitable for a certain type of a plant, the geometry shader generates a simple vegetation geometry with correct texturing information. Otherwise no geometry is being generated. Some work could be done in the vertex shader, but this would introduce additional data transfer between shaders.

The dynamic texturing also brings the possibility of altering the whole terrain look by simply changing a few values. The whole terrain can change from snow covered arctic scene into grassy hills or sandy dunes very easily in real-time, as Figure 2 illustrates.

2.3 GPGPU

It is easy to map cellular automaton to the GPU. It can be achieved even by the classical graphics API like OpenGL with GLSL shaders, which is favorable in this case, because the code is highly portable and we do not have any problems with visualization of the simulation results.

The 2D cell grid data can be easily stored in a few 2D textures with `GL_NEAREST` interpolation. We can have one RGBA texture for the terrain height, water column, water column average, and water sediment dissolved in the water. The water speeds for 4 directions can be easily stored in another RGBA texture.

Since we need to compute the next state from the previous, we need two sets of these textures to switch between, reading from one and writing with multiple render targets to the other. Precision is also important. The current hardware provides various fixed or floating point data types. The best solution is to use 32-bit floating point for a component. 16-bit floating point types are also possible, but the precision for some operations is sometimes too low.

3 RESULTS

The first implementation was done with the bechelor's thesis. It was CPU based with only the water simulation. The CPU had to compute new cell states, build new geometry and compute surface normals. Then all was transferred to the GPU, producing a significant overhead, and rendered.

During the master's thesis, the whole program was remade to run the simulation on the GPU. Dynamic texture mapping, vegetation and erosion were added. Since it maps extremely well on the highly parallel GPU hardware and the data transfer overhead was removed, the speedup against the simpler CPU version was about two orders of magnitude.

The current main limitation is the GPU memory size. GPU can easily handle faster than real-time computations over textures fully occupying available resources. The rendering is much more demanding due to the difficulties with proper rendering of the water geometry layered over the terrain and resulting problems with level of detail implementation.

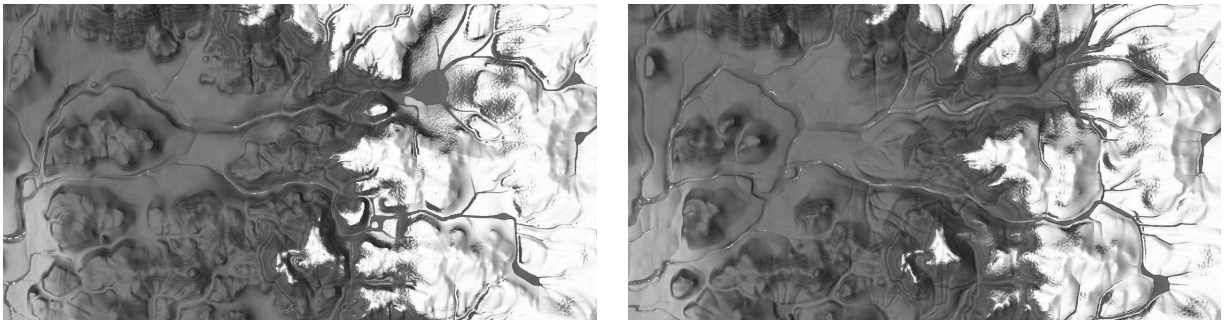

Figure 1: The advance of simple erosion.

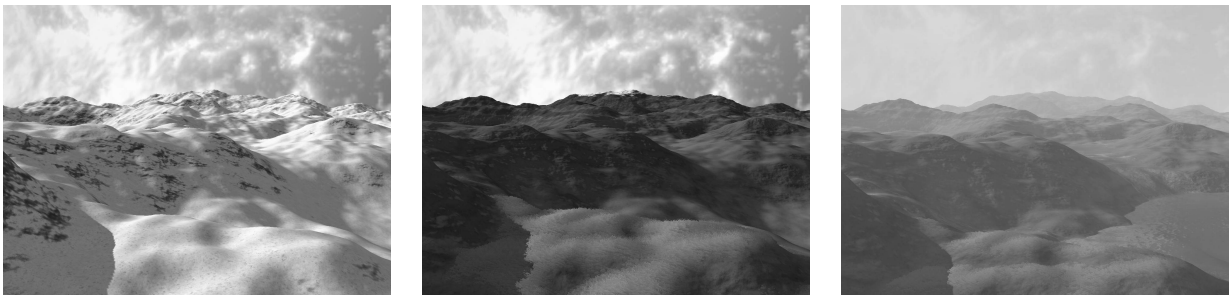

Figure 2: The various texturing parameters and water state in the same terrain.

There is a video avaliable at http://www.youtube.com/watch?v=6uNls7qqPTQ

4 DISCUSSION

In the master's thesis, there were several additional effects implemented, which can not be discussed here in depth. The most important are: procedural skybox dynamically lighted in real-time, height based fog with simple ray tracing, and texture mapped shadows.

Several directions were proposed for the next development, however some of them, like multi-layer terrain and connecting more cellular automatons into one system were subject of another work [7]. This work differs mainly in the orientation on the graphical part of the problem and the interconnection with dynamic texturing and vegetation generation.

In the future work, the level of detail problem should be addressed. Some texturing parameters and vegetation information could be also given by a texture, possibly with a lower resolution than the base terrain, producing a larger diversity in the natural scenery. More complex geometrical object should be also examined.

5 CONCLUSION

This work confirms, that dynamic large scale real-time water simulation is no longer impossible and it does not necessarily have to consume all the available resources. The dynamic water significantly improves the realism of some scenes and makes more scenes possible. The real-time erosion is useful only in a limited range of scenes like floods or tsunami, however the potential of aiding in terrain design tools or even scientific applications is clear. While this work was mainly targeted on the speed and graphics, more complex and realistic shaders can be used for various purposes. The simplicity of integrating water simulation and erosion with terrain texturing and automatic vegetation generation is also shown, demonstrating the potential for more complex systems.

ACKNOWLEDGEMENT

This work was partially supported by the BUT FIT grant FIT-S-10-2 and the research plan MSM0021630528.

REFERENCES

[1] Müller, M., Charypar, D., Gross, M.: Particle-Based Fluid Simulation for Interactive Applications. Eurographics / SIGGRAPH Symposium on Computer Animation 2003

[2] Krištof, P., Beneš, B., Křivánek, J.: Hydraulic Erosion Using Smoothed Particle Hydrodynamics. Eurographics 2009, vol. 28 No.2

[3] Crane, K., Llamas, I., Tariq, S.: Real-Time Simulation and Rendering of 3D Fluids. In GPU Gems 3, ch. 30, Addison-Wesley

[4] Vlček, A.: A Simple Simulator of Running Water in Real-Time Using OpenGL. FIT BUT. Advanced Assembly Languages Project 2005.

[5] Vlček, A.: Water Flowing in the Landscape Visualization. Brno, 2007. FIT BUT. Bechelor's thesis, supervisor Seeman, M.

[6] Vlček, A.: Real-Time Weather in a Landscape Visualization. Brno, 2009. FIT BUT. Masters's thesis, supervisor Seeman, M.

[7] Šťava, O., Beneš, B., Brisbin, M., Křivánek, J.: Interactive Terrain Modeling Using Hydraulic Erosion. Eurographics / ACM SIGGRAPH Symposium on Computer Animation 2008