

# Height Field Water Simulation

Joseph Wilson, Arnaud Filliat, Jacob Davis

May 9, 2017

# **1 Abstract**

Provide a one paragraph summary of your entire project. Very briefly mention why this technique is important, what makes it challenging, what your solution was, and what your results were. This is essentially condensing each section listed below into one or two sentences. You may want to actually write this section last.

# **2 Introduction**

Give a high level overview of your project. Why is it important? What makes it challenging/difficult? Describe what will follow in the rest of the report. This section should be several paragraphs long (roughly one plus column).

# **3 Related Work**

Discuss what has previously been done in this area. What existing and similar solutions exists? What have other people done? Where did those existing solutions fall short? You should be citing your references the most in this section (though you will probably need to cite in the first three or four sections). The length of this section will vary.

# **4 Problem Statement**

Go into technical detail about the challenges in this area. What specifically makes it difficult? What shortcomings did other techniques have?

# **5 Problem Solution**

Any water simulation has two primary parts, the water movement and the water lighting. Our implementation focuses on water movement. The height field information is primarily stored in two, two dimensional arrays. One for column height and one for column velocity. These columns are updated each frame. Each column calculates the force on it due to the four surrounding columns with the equation,  $f = c2*(u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1]-$

$4u[i,j])/h^2$ .  $C$  represents the speed of a wave and  $h$  represents the width of a column. Our simulation takes place on a large scale with each column representing a 1 meter by 1 meter column of water. After the force is calculated it is applied to the velocity of each column. The force is multiplied by the amount of time that has passed since the last update,  $1/60$  seconds in our case. This ensures that steps remain constant even if framerate is changed. After all velocities have been updated the height of each column is modified by adding velocity multiplied by the time step. This process of forward Euler integration will lead to feedback loops if not controlled so we implement damping and clamping. Velocity is damped by a factor of 0.99 each step and column height is bounded between -100 and 100 although with the damping factor these extremes are rarely reached.

An important part of the simulation not yet discussed is boundary conditions. . . .

Object interaction is implemented with a cube. We chose to opt for a simple implementation and our cube is therefore drawn with edges parallel to the edges of our columns. During each step of the simulation we find the location of the corners of the cube with respect to the indices of our columns. This allows us to compare the cubes height with that of the water. By subtracting the height of the base of the cube from the height of the water and clamping between 0 and cube width we calculate the amount of water displaced by the cube. The displacement of each column is stored in another 2-dimensional array and by comparing the new displacement with the old we are able to calculate the change in displacement of water by the cube. Once the total displacement and the change in displacement are known these values are used to calculate the force on the cube and the change in water level around the cube respectively. Any displaced water is added directly to the columns surrounding the cube. This also occurs when an object is removed with water being subtracted from the surrounding columns. Our height field inside the cube is not modified allowing water to cover the cube should it fall below the water level. Because the height field is not modified on the inside of the cube a feedback loop can occur. The displaced water is added to the surrounding columns which flow inward increasing the amount of displaced water by the cube, adding more water to the surrounding columns. This was solved in two ways. The first option sets the height of each internal column to halfway up the cube. This allows the following update due to column velocity to pull a column above the cube and prevents the feedback loop but gives the water a much more viscous appearance with it

clinging to the cube, creating depressions and mounds in the surface when the cube is dropped and raised respectively. The second option is to use the average water level instead of the local column height. This leads to realistic simulations when the water is relatively flat but leads to disconnected water ripples and cube movement if the surface is far above or below average at that location. Once the water is updated the cube is moved using Forward Euler integration and standard physics equations. Force due to water is calculated with  $\text{density} \times \text{gravity} \times \text{displacement}$  and both it and gravity are applied to velocity and then velocity to cube location.

It is also important to ensure water is not added or removed from the simulation. . . .

The final step is drawing the height field. To do this we created an additional array that contains a point representing the upper lower left corner of each column. The x and z components were the indices of a location and the y the height. This array was transferred to the GPU via VBO and VAO to as points. A geometry shader then took each point and expanded it into a column, adding sides with specified widths and a height determined by the height of each point. The columns were then shaded with Blinn-Phong shading. A simple set of Blinn-Phong shaders were applied to the cube as well.

## 6 Results

How did your implementation work? Did it accomplish what it set out to do?

## 7 Conclusion

In a couple paragraphs, summarize all of the above sections. Give any final thoughts on lessons learned, challenges that arose and how they were overcome, and next steps to take to future work.

## References

- [1] Daniel Mikeš. “Real-time Water Simulation with Wave Particles on the GPU”. In: (2015). URL: <http://old.cescg.org/CESCG-2015/papers/>

Mikes-Real-time\_Water\_Simulation\_with\_Wave\_Particles\_on\_the\_GPU.pdf.

- [2] Bálint Miklós. “Real-Time Fluid Simulation Using Height Fields”. In: (2004). URL: [http://www.balintmiklos.com/layered\\_water.pdf](http://www.balintmiklos.com/layered_water.pdf).
- [3] Matthias Müller-Fischer. *Fast Water Simulation for Games Using Height Fields*. 2008. URL: <http://matthias-mueller-fischer.ch/talks/GDC2008.pdf>.
- [4] roxlu. *Height Field Simulation on GPU*. 2013. URL: <http://roxlu.com/2013/024/height-field-simulation-on-gpu>.
- [5] Barbara Solenthaler et al. “SPH Based Shallow Water Simulation”. In: (2011). Ed. by J. Bender, K. Erleben, and E. Galin. URL: <http://matthias-mueller-fischer.ch/publications/SPHShallow.pdf>.
- [6] Zejia Zheng. *Height field fluids simulation and shallow water equation*. URL: <http://zejiazheng.com/?p=87>.