

Reducing Ineffectual Computations in Water Simulation

Jason Thompson

La Trobe University, Bundoora, Victoria, Australia
Faculty of Science, Technology & Engineering
Department of Computer Science and Computer Engineering
j9thompson@students.latrobe.edu.au
<http://www.latrobe.edu.au/cs>

Abstract. This paper proposes an abstract water simulation system composed of a low cost global simulation which is further refined in a view-dependent fashion by a local simulation. The global simulation is explored with a focus on identifying and reducing ineffectual computations. Test results indicate in certain scenarios that the global simulation has a decreased number of computations with a similar distribution of water compared to an un-optimised version.

Key words: optimisation, water, simulation, fluid, region, height-field

1 Introduction

This research focuses on reducing the amount of computations in a water simulation. An abstract system is proposed that separates a water simulation into a two components: A global simulation which is performed all over a spatial domain and a local simulation which takes the results of the global simulation and produces a more detailed view dependent representation. The global simulation is the focus of this paper along with the optimisations that can be applied to it, namely to stop simulating regions of water that will not change as well as an approximation of waterfalls.

Section 2, related work, analyses other ways large scale water simulation has been accomplished. Section 3, methods, describes the abstract system, as well as the implementation of the global simulation component. Section 4, results, presents and discusses the results of the implementation. Finally, Section 5, conclusion, presents the conclusion of the research as well some of the potential areas for future work.

1.1 Problem definition

There are a diverse number of water simulation techniques that can produce real-time results, however, these approaches do not scale well to large sizes. The time it takes to compute grows too rapidly. When looking at how large scale water has

been represented, two main approaches become apparent. The first is a static global representation of water and a dynamic local representation, an example of which is, provided by Hinsinger et al. [2] and Johanson [4] who generate procedural waves only on visible sections of pre-defined water. The second is a dynamic global representation of water. An example of this is provided by Maes et al. [8] who use a height-field to represent water. The first approach is generally computationally cheaper, but static. The second approach is generally too computationally expensive to be feasible for large scales. The problem is then how can the two approaches be combined to create a simulation that is somewhere in between the two extremes in terms of dynamism and computation time.

1.2 Summary of contributions

- An abstract system to cheaply simulate water over a terrain.
- Two optimisations:
 - An approximation of waterfalls which reduces the number of particles.
 - A way to identify and cull ineffectual computations at a coarse granularity.

2 Related work

2.1 Basic Approaches to Water Simulation

Large scale water simulations generally use similar methods to smaller scales of simulations, but apply optimisations to make the new approach feasible. So it is useful to first provide a short review of the basic approaches to water simulation.

Procedural Models Procedural methods do not attempt to do any form simulation to generate a fluid. Instead they “simulate the effect, not the cause” Muller[12]. The effect of a physical phenomenon is generated using variables that describe the different properties of the effect. For example, in the case of an ocean simulation, properties like the choppiness of water, the depth of the water and the wind conditions might be used. Tessendorf [16] provides an overview of two of the main procedural methods: “Gerstner Waves” and “Statistical Ocean Sampling”. “Gerstner Waves” aim to simulate an ocean scene by summing two-dimensional sinusoids while “Statistical Ocean Sampling” uses the Inverse Fourier Transform to synthesize an ocean surface from a procedurally generated frequency spectrum.

Height-field Methods One of the earliest Computer Graphics papers which modelled water as a height-field was Kass and Miller [5] which used a simplification of the Navier-Stokes equations to determine the flow of water between water columns. O’Brien et al. [14] took a different approach and used the concept of

hydrostatic pressure. ‘Virtual Pipes’ are constructed to and from adjacent water columns. Using the hydrostatic pressure between the ‘virtual pipes’, the flow of water between columns is calculated. [14] was also able to simulate two-way rigid body coupling through the use of a separate surface sub-system. This allowed rigid bodies to apply forces on the free surface of the water and for the water to apply forces on the rigid body. Both approaches have been extended recently [7][17].

Eulerian Grid Methods Eulerian grid based methods are characterised by the assumption that a fluid can be represented by a vector field. Usually this assumption results in a two-dimensional or three-dimensional grid with velocity, forces and pressures defined at discrete points. It has been used extensively in scientific and engineering domains. However, because of its high memory and processing requirements it has not been used as much in real-time applications. That being said, Stam [15] presents a Semi-Lagrangian two-dimensional algorithm for the simulation of fluids in real-time. Though the algorithm has Lagrangian aspects, the result of the algorithm is still Eulerian, a velocity field. The algorithm convects a density field through an evolving velocity field. The demonstration application has the appearance of smoke moving in air.

Particle Systems Particles systems are a flexible tool for simulating physical phenomena and have been used in many different areas, more relevantly in fluid simulation. There are two main types of particle system used in fluid simulation: non-interacting particles where particles do not interact with each other and interacting particles where particles interact with each other. A number of papers [14][7][8] use non-interacting particles to model when water disconnects from itself, for example when water splashes. Mass conserving particles would be emitted when the upward velocity of the water reached a critical amount. On returning to the water they would be reabsorbed. Particles used in this fashion have the advantage of being simple and efficient.

While simple and efficient, non-interacting particles can only represent a limited number of phenomena such as spray, splashes and/or waterfalls. On the other hand, interacting particles can be used to represent more general cases like water being poured into a cup. This generality comes with a cost, that is, interacting particles generally require a lot of computations to determine particle-particle interactions. One of the earliest computer graphics techniques put forward was by Miller and Pearce [9] where each particle applies attractive, repulsive and drag forces on every other particle within a defined distance. More recently, Müller et al. [13] put forward a real-time physically based approach using concepts borrowed from Smoothed Particle Hydrodynamics (SPH) and fluid dynamics. One of the main bottlenecks of this approach is the extraction of a renderable surface.

2.2 Large Scale Water Simulation

These approaches have been modified to accommodate large scale simulations. Some of the main issues with large scale water simulation are:

Creation of the large water system Most systems that simulate the dynamic creation of water systems have a similar high level view. The system takes in an arbitrary terrain, sources and sinks of water and then solves the distribution of water over finite time steps. What makes this so challenging is that real-time performance must be maintained even though there is a large spatial domain and large amounts of water.

Kipfer et al, [6] uses SPH to simulate water and as a result the computational domain is directly related to the amount of water in the simulation. Chen et al. [1] uses two systems one to track the footprint of water and another to perform a height-field simulation on that footprint. The computational domain is related to the surface area of the water. Maes et al. [8] uses a height-field simulation over the entire spatial domain but adapt their algorithm to run on a GPU and were able to produce real-time results.

Generation of surface detail Surface detail is an important part of simulating large bodies of water. It adds texture to the visual appearance with features like ripples, swells and waves. It also has the important role of providing information on the type and state of the water. For example, to indicate that a section of water belongs to a stormy ocean the surface details will be large choppy waves. The goal of surface details creates some challenges: How to produce meaningful surface details whilst maintaining real-time performance. Yu et al. [18] generate surface details for a flowing river using advected particles representing samples of wave textures. Whereas, Mitchell[10] uses a GPU implementation of procedural waves to provide surface detail for an ocean scene at real-time rates.

Efficient display of large water systems Rendering large water systems is very time consuming because of the amount of rendering that must be done and the cost of rendering itself. It is not always feasible or desirable to display the entire scene, so view dependent sub-set is rendered in place of the entire scene to reduce unnecessary rendering operations. This notion has been applied to unbounded water by Hinsinger et al. [2] who projects a water surface from screen space to world space which ensures only visible water is rendered. Johanson [4] provides a more thorough treatment of a similar projection scheme. Hu et al. [3] takes a similar approach but restricts the real-time generation of waves to water close to the viewer.

3 Methods

I would like to propose an abstract system which will allow for more efficient simulation of water on a large scale at real-time rates. The system is structured

into four main parts: the physical volume model, the physical volume model manager, the local simulation and finally the renderer. The goal of this structure is to have a low cost global simulation composed of the physical model and physical model manager that produces a rough approximation of the water. As well as a local simulation that takes the rough approximation of the water and produces a more detailed view-dependent representation of the water that can be rendered.

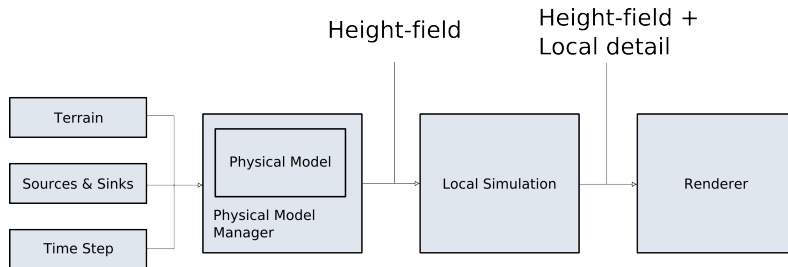


Fig. 1. A conceptual view of the proposed system.

Given the limited time available and other factors, only the two-dimensional case is looked at. This then removes the need to do any complicated rendering and hence the renderer will be omitted. In order to test the concept of the system, a physical volume model must be selected from those that have been described in the Section 2. The simplest approach will be used to test that this system is viable. If the concept proves viable later iterations can focus on applying more advanced algorithms. The physical volume model must have the following qualities:

- The approach must be able to produce results at a real-time rate. If the simulation cannot produce real-time results then no amount of selective application will produce real-time results.
- The approach must be able to simulate the volume and flow rates of water in space and time.
- The approach must be a physically based model of water.

A height-field approach appears to be the best fit. From the different height-field models reviewed, the Pipe and volume model[14] presents many advantages. One being that it is able to be coupled with particle systems to handle discontinuities in water. Another being that it can be integrated over time with a simple numerical integration scheme. Alternatively, a Navier-Stokes equation approach can provide a more realistic water simulation; however, at the expense of a more complicated integration scheme which involves solving a system of linear equations. For this paper, the pipe and volume model appears to be the ideal choice.

3.1 The Physical Volume Model

The physical volume model is based on the work of O'Brien et al. [14] which was later extended by Mould and Yang [11] amongst others. A very brief overview of the model is reproduced here, a more detailed description can be found in the mentioned sources. The model represents the terrain as a height-field on top of which a number of columns are constructed. Columns are connected to their neighbours by pipes which allow water to flow from one column to its neighbours. The system simulates water by calculating the pressure at either end of a pipe and hence the flow rate across it.

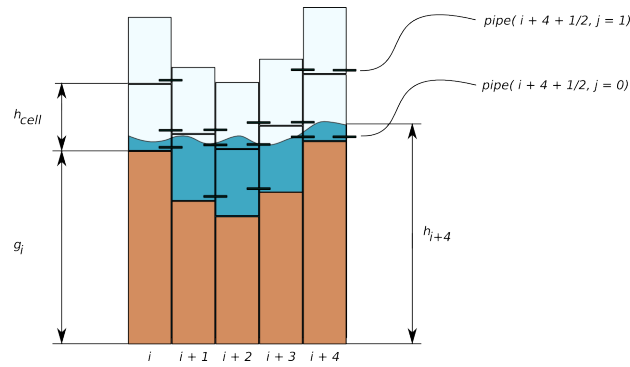


Fig. 2. The height-field model

First the terrain is defined as a height-field where every spatial coordinate corresponds to one height value. The height of the terrain is g_i where $i \in \mathbb{N}_0$ represents the spatial index. At each point i on the terrain a column is defined which holds a volume of water v_i^t , note that the superscript is not an exponent but a time index. Using the volume of water in a column the water height, h_i^t at column i , can be determined with:

$$h_i^t = \frac{v_i^t}{w} + g_i \quad (1)$$

Where w is the width of the columns. A column's change in volume and hence change in height can be determined with:

$$v_i^{t+\Delta t} = v_i^t + \Delta t \eta_i^{t+\Delta t} \quad (2)$$

Where $\eta_i^{t+\Delta t}$ is the flow rate into column i at time $t + \Delta t$. It is now possible to calculate the heights of water on the terrain over time.

Particle System and Sources There are two more structures that must be defined sources/sinks and the particle system. The particle system is used to

model discontinuities. Particles are emitted from a water column when the height difference of columns is greater than a threshold. Each particle carries some mass, $m_{particle}$, and is acted upon by gravity and air friction:

$$\vec{F} = \vec{g} - f_{drag} \vec{v} \quad (3)$$

Where $f_{drag} \in (0, 1]$ is the drag coefficient. When a particle comes in contact with the height-field its mass is reabsorbed and an ad-hoc force is applied to the water, $F_{collision} = v_y m_{particle} c$, where v_y is downwards component of the particle's velocity and c is constant that is used to tune the results. Now let a particle be defined as a triple $p = (\vec{x}_p, \vec{v}_p, m_p)$ where \vec{x}_p is the particles position, \vec{v}_p is the velocity of the particle and m_p is the mass of the particle.

Sources and sinks add/subtract a certain amount of water into/from a particular column over some time. Traditional approaches have fixed the height of a water column to indicate a source or sink. However, this dampens or eliminates ripples that propagate over a source column. So a different approach is to introduce a certain amount of water every time step, analogous to a pump. Let a source be a pair $s = (\eta_s, j_s)$. Where $\eta_s \in \mathbb{R}$ is the sources flow rate and j_s is the index of the destination column. To update a source s we add its contributing volume $\Delta t \eta_s$ to its destination column v_{j_s} . The set of sources and sinks is given by $S = \{s_1, s_2, s_3, \dots, s_{n_{sources}}\}$.

3.2 The Physical Volume Model Manager

The fundamental unit used in the model manager is the *region*. A region contains a set of adjacent columns and a set of sources and sinks. Regions have the condition that they do not overlap each other. Let H be the set of all water column heights $H = \{h_1, h_2, h_3, \dots, h_n\}, h_i \in \mathbb{R}_{\geq 0}$. From the previous section we have the set S of all sources. We have a set of the terrain heights $G = \{g_1, g_2, g_3, \dots, g_n\}, g_i \in \mathbb{R}_{\geq 0}$. So now a region can be defined as a triple $r = (H_r, S_r, G_r), H_r \subseteq H, S_r \subseteq S$ and $G_r \subseteq G$.

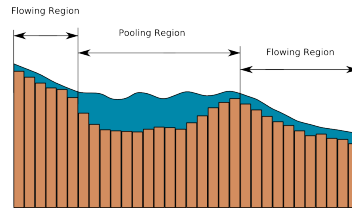


Fig. 3. An overview of the region model

Regions are created, destroyed and merged throughout the simulation. The first region, a pooling region, is created when a source of water is added to the terrain. This region grows until it comes in contact with a downward slope. At

which point, a flowing region is created to contain the water as it flows down the slope. Alternatively, if the terrain is steep enough, a waterfall region is created. The behaviour of each of these is described in section 3.2.

How ineffectual computations can be reduced can now be described. In a region there will be a number of inputs/outputs of water such as sources, waterfalls and an inflow and/or outflow of water from/to adjacent regions. If the assumption is made that the volume model is primarily concerned with the transport of water and not necessarily the shape of its surface then an optimisation can be made when the net flow of a region is zero. At that point, the simulation of that region can be stopped as that region's volume will remain constant and will no longer grow or shrink. So formally, the net flow of a region, r , is defined by:

$$Netflow(r, t) = \sum_{s \in S_r} \eta_s + \eta_{left}^t + \eta_{right}^t \quad (4)$$

Where η_s is the flow of an individual source, η_{left} is the flow rate at the leftmost column of the region and η_{right} is flow rate at the rightmost column of the region. The volume in a region R_i at time t is $Volume(r, t)$ and the volume at time $t + \Delta t$ is:

$$Volume(r, t + \Delta t) = Volume(r, t) + \Delta t Netflow(r, t) \quad (5)$$

Now if the $Netflow(r, t) = 0$, it follows that $Volume(r, t + \Delta t) = Volume(r, t)$. The simulation can be stopped in that region as its volume will not change and is considered stable. However, in practice this condition is not usually met; the surface waves of a region create a variation in the flow rate. In order to reduce this noise, two factors are introduced: A constant $C_{stable} \in \mathbb{R}_{\geq 0}$ the magnitude of the net flow needs to be below this threshold to be considered stable; however, introducing this constant will also introduce an error either a loss or an increase of volume to the system as shown in the results, Section 4. The second factor is to take a moving average of the net flow over time. This eliminates sudden sharp increases or decreases of the surface from affecting the stability. The average net flow is given by:

$$avg(Netflow(R_i, t)) = \frac{1}{n_{samples}} \sum_{j=0}^{n_{samples}} Netflow(R_i, t - j\Delta t) \quad (6)$$

Where $n_{samples}$ is the number of samples. We assume that Δt is constant and $t > j\Delta t$ this means that we choose not to calculate the average net flow and hence stability of a region until we have $n_{samples}$. So the condition for stability now becomes: A region r_i is stable at time t iff $|avg(Netflow(r_i, t))| < C_{stable}$

Regions In this system regions are updated and as a result the columns they contain. At every simulation step, each region looks at its boundaries, the height and contents of the adjacent columns. It will then determine if any action should be taken. For example, if a pooling region's boundary contains water and the adjacent column is empty, the region will extend outwards.

TerrainType(p)	Region Operation
uphill	extend region
downhill	create a flowing region
sharp drop	create a waterfall region
boundary	Nothing
pooling region	merge two regions
flowing region	extend region
waterfall region	connect regions

Table 1. Pooling Region Behaviour

TerrainType(p)	Region Operation
flat	extend region
uphill	create a pooling region
downhill	extend region
sharp drop	create a waterfall region
boundary	create a pooling region
pooling region	connect regions
flowing region	connect regions
waterfall region	connect regions

Table 2. Flowing Region Behaviour

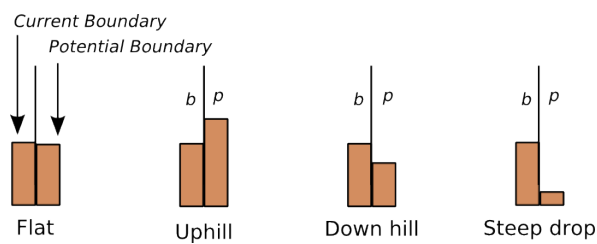


Fig. 4. The classification of columns

There are three types of regions: pooling, flowing and waterfall. Each region is differentiated by the way they its boundary changes. Table 1 and 2 give a description of the behaviour of the pooling and flowing regions respectively. To determine how the boundary of a region should change there needs to be some understanding of the potential new boundary. So it is classified into a number of different types. Let b be the index of the left or right boundary of a region and p be the potential new boundary of that region where $p = b \pm 1$. The function $TerrainType(p)$ takes in the potential new boundary and returns its classification (see figure 4 for some sample classifications).

Water Fall Region The waterfall region differs from the flowing and pooling regions in that it contains only one cell and it does not transfer water to its neighbours but through the air see figure (a) 5.

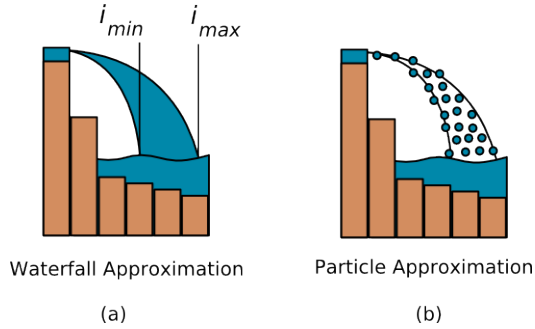


Fig. 5. The different types of waterfalls

To determine where the water will be transferred to, two particles are emitted representing the furthest a particle would travel p_{max} and the closest a particle would travel p_{min} from the waterfall region. The initial velocity is given by:

$$\vec{v}_{max}^{\rightarrow t_0} = \vec{i} (1 + C_{x_{splay}}) \eta_j^t + \vec{j} \frac{dh_j}{dt} \quad (7)$$

$$\vec{v}_{min}^{\rightarrow t_0} = \vec{i} (1 - C_{x_{splay}}) \eta_j^t + \vec{j} \frac{dh_j}{dt} \quad (8)$$

The path of both particles are traced until they make contact with the terrain at i_{max} and i_{min} . Where these two particles collide with the ground, a new source and region are added. The source added differs from the previous type because it transfers water to many columns. Also its inflow is the outflow of the waterfall region. If there is already a region where the particles land, then the source is added to that region otherwise a new region is formed. This region is continuously extended to encompass the points between i_{max} and i_{min} inclusively.

4 Results

The tests aim to determine if computations are culled and to identify the types of errors introduced by the optimisation. Each test has one source initially, s_0 , and a sink, s_1 , of equal magnitude is added after some time. The expected results for the optimised version are that the total volume increases until the sink is added. At which point, the total volume will remain constant and soon after all regions will stabilise and no computations will be performed. Ideally, the difference in volume between the optimised and un-optimised versions, or the error, should be small and not grow over time. During every simulation step the following metrics will be recorded: Total column volume and flow updates, total volume of the system and total number of particle position updates.

4.1 The Downhill Case

This scenario tests the ability of the algorithm to eliminate ineffectual computations on a hill. In figure 4.1 $\eta_{s_0} = 3$, $\eta_{s_1} = -3$ and $C_{stable} = 0.0005$. The threshold is too small and the system is unable to stabilise; the performance is bounded by the un-optimised version.

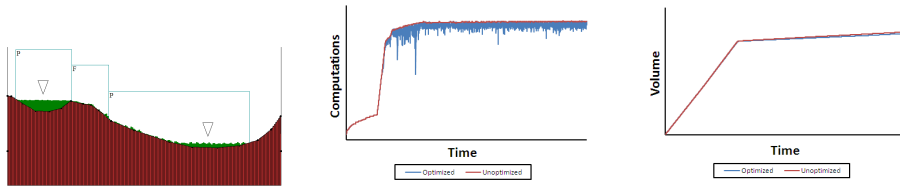


Fig. 6. Unsuccessful results of the downhill case

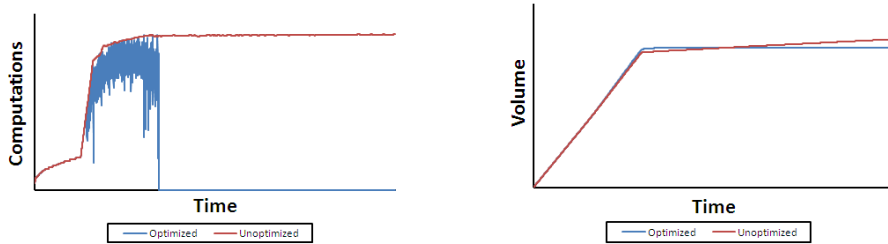


Fig. 7. Successful results of the downhill case

A better threshold value can be determined which will produce the desired behaviour. In figure 4.1 $\eta_{s_0} = 6$, $\eta_{s_1} = -6$ and $C_{stable} = 0.007$. The number of

computations behaves as expected; however, the error grows over time as can be seen from the continued increase in difference of volume. A number of issues have become apparent from this test:

- Inaccuracies in the underlying physical simulation affects the results. The chart shows that the un-optimised system’s volume keeps growing after the sink has been added. This affects possible measures of accuracy.
- The selection of a stabilisation threshold is an ad-hoc process. A number of different values have to be attempted before a successful one can be found.
- If the stabilisation threshold is too close to the source’s value, the region will stabilise because the inflow is considered to be non-existent.

4.2 The Step Case

This tests to see if the volume approximation of a waterfall is more efficient than a particle approximation, see figure5 for the difference. In this test $\eta_{s_0} = 6$, $\eta_{s_1} = -6$ and $C_{stable} = 0.0005$. This was the most successful test.

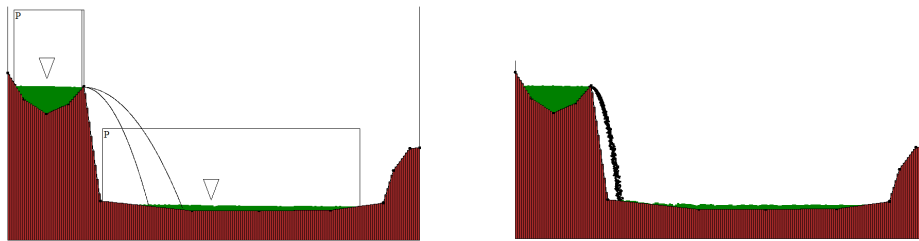


Fig. 8. Volume approximation of a water fall (left). Particle approximation of water fall (right)

The computations dropped and the error did not increase over time. In terms of the number of particle updates, the optimisation performed less updates in the long term but more in the short term. This is because the path of the waterfall must be traced, for example an unstable region could perform 1000 particle updates for tracing the path, whereas the un-optimised version may only have 300 particles and would perform 300 particle updates.

5 Conclusion

There is a large amount of work done on fluid simulations for Computer Graphics in real-time and off-line. After evaluating a number of simulation techniques for management, a height-field approach was selected. Using the inflow and outflow of water into a region, ineffectual computations were able to be reduced. Test results show that this type of optimisation approach has potential with certain

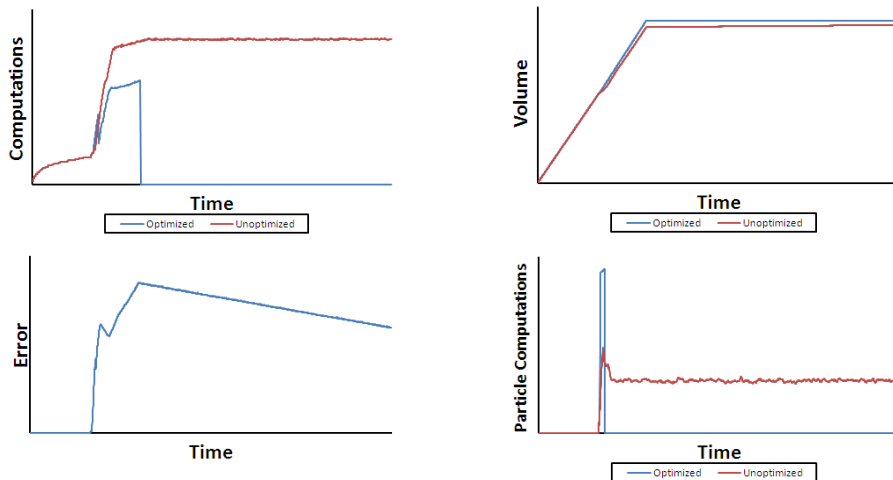


Fig. 9. Results of the steep case

cases showing decreased computations with comparable results to the existing technique being managed. A number of problems were encountered which must be solved before this approach can be considered a robust and useful method. Those being the cumulative error in the flow rate of regions created by reducing computations and the difficulty in determining when a threshold for when a region should be considered stable.

Some possible directions for future work include finding alternate metrics for identifying ineffectual computations. Another is extending and testing the approach in three-dimensions to see if the concepts carry forward. Lastly integrating the system with a local simulation to represent waves and particles in stabilised regions. With further studies this approach can be a useful addition to real-time water simulation.

References

1. Jim X. Chen, Niels da Vitoria Lobo, Charles E. Hughes, and J. Michael Moshell. Real-time fluid simulation in a dynamic virtual environment. *IEEE Comput. Graph. Appl.*, 17(3):52–61, 1997.
2. Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive animation of ocean waves. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 161–166, New York, NY, USA, 2002. ACM.
3. Yaohua Hu, Luiz Velho, Xin Tong, Baining Guo, and Harry Shum. Realistic, real-time rendering of ocean waves: Research articles. *Comput. Animat. Virtual Worlds*, 17(1):59–67, 2006.
4. Claes Johanson. Real-time water rendering - introducing the projected grid concept. Master's thesis, Department of Computer Science, Lund University, 2004.

5. Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 49–57, New York, NY, USA, 1990. ACM.
6. Peter Kipfer and Rüdiger Westermann. Realistic and interactive simulation of rivers. In *GI '06: Proceedings of Graphics Interface 2006*, pages 41–48, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.
7. Marcelo M. Maes, Tadahiro Fujimoto, and Norishige Chiba. Efficient animation of water flow on irregular terrains. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 107–115, New York, NY, USA, 2006. ACM.
8. Marcelo M MAES, Tadahiro FUJIMOTO, and Norishige CHIBA. Low-memory and interactive-rate animation of water-column based flows. *The Journal of the Society for Art and Science*, 7(1):1–13, 2007.
9. Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.
10. Jason L. Mitchell. Real-time synthesis and rendering of ocean water. In *ATI Research Technical Report*. Marlboro, MA, 2004.
11. David Mould and Yee-Hong Yang. Modeling water for computer graphics. *Computers Graphics*, 21(6):801 – 814, 1997. Graphics in Electronic Printing and Publishing.
12. Matthias Müller. Real time fluids in games. <http://www.matthiasmuller.info/talks/gameFluids2007.pdf>, 2007.
13. Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.
14. J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation*, page 198, Washington, DC, USA, 1995. IEEE Computer Society.
15. Joe Stam. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference, March 2003*, 2003.
16. Jerry Tessendorf. Simulating ocean water. In *SIGGRAPH 2004 Course Notes 31*, 2004.
17. Huamin Wang, Gavin Miller, and Greg Turk. Solving general shallow wave equations on surfaces. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 229–238, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.
18. Qizhi Yu, Fabrice Neyret, Eric Bruneton, and Nicolas Holzschuch. Scalable real-time animation of rivers. *Computer Graphics Forum (Proceedings of Eurographics 2009)*, 28(2), mar 2009. to appear.