Simulating Wave Particles

Jonathan Cheng and Dhruv Patel
Rensselaer Polytechnic Institute of Technology

Abstract

After learning of fluids and different methods of simulation, we were interested in other realistic techniques of simulation. The principle in simulating fluids is modeling a solution to the Navier-Stokes Equations. These equations represent the mathematics behind incompressible fluids with aspects like pressure and viscosity. From our time working on homework 2, we have had experience in working with Eulerian grid based solutions to the Navier-Stokes equations [Foster and Metaxas, 1996].

In this paper, we decided to look into and implement one of the many techniques used to model and simulate fluids, specifically liquids since gasses are considered fluids. We specifically look at Wave Particles [Yuksel et al, 2007], which are used to model the surfaces of fluids and can be used to model interactions with fluids.

1 Introduction

Fluid simulation and modelling has been a continuous field of study throughout the past decade in computer graphics. Many papers document new models for simulation improving upon detail and realism while maintaining speed. For this final project in the class we looked at different ways to model fluids and decided on trying to implement Wave Particles, a nice 'hack' at simulating the waves made by liquids and the surface interactions fluids have. Another method of modeling we discovered called Smooth Particle Hydrodynamics (SPH) [Ihmsen, 2014] was also considered for this project but we scrapped the idea due to scale and disconnection with Wave Particles. Our goal was to simulate realistic fluid surfaces and their interactions using Wave Particles.

1.1 Overview

Our approach to use Wave Particles stemmed from the complexity and realism they provided. The concepts present by Yuksel et al. in his paper were simple in representing surfaces of liquids, but the underlying implementation was challenging to say the least. While techniques like SPH are meshless, they require many points of data and are computationally complex on the CPU, Wave Particles are fast and realistic enough for modelling simple waves and fluid interactions. Although there have been improvements made to SPH and its computation complexity, Wave Particles are much faster and more reactive to interactions.

In the next section we describe the implementation and our code to apply Wave Particles. In section 3 we analyze our results and shortcomings on this project. We conclude with section 4 and discuss further areas of improvements.

2 Implementation

Initially we planned to extend the code used in homework 2, with the Eulerian grid based solution, by adding wave particles on the surface. This extension was not feasible and we scrapped our attempts after trying for several hours. This meant we had to implement wave particles without any previous code, in essence starting from scratch. We were able to use our homework 2 as reference and even utilized the renderer and shaders for our project, but the core implementation was made for our specific purposes. Like homework 2, we implemented in project in C++ and utilized OpenGL for rendering our simulations. We also used the vertex and fragment shaders as well as the renderer files and the initial framework for input provided from our homework 2 code.

2.1 Initial Attempts

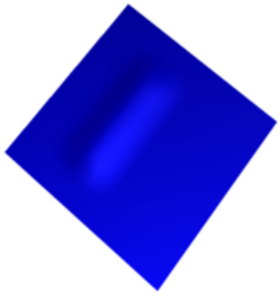

Figure 1 Simple Wave From Wind

As stated before, our initial attempts at extending the previous homework files wasted a lot of our time. We assumed that the Eulerian grid based method could be extended with wave particles without much changing but we were incorrect. Previous work had indeed reinforced the idea of extending the volume based method used in homework 2 with a surface model and even extended to a spray model for splashing [O'Brien and Hodgins, 1995]. This model and paper gave us the idea to implement wave particles for the surface model portion. However, our attempts at combining and extending our grid based volume model into a column based model and adding surface particles ended up being useless. These attempts at extension required too many changes and revisions to the already existing code, we deemed it better to start from scratch.

Therefore, we decided to start with our surface model, wave particles, and scaled the scope of this project down to only focusing on the surface and interactions with it. This lead to the current work in progress we have on wave particles. The process of starting over began with new classes for each element in the scene. We made classes for the wave particles, the wave itself (i.e. the surface), and the objects interacting with the surface. Each object was further separated into their own class for convenience. In total we had, four new classes representing our scene, one for the surface, one for the wave particles, and one for each object which were a sphere and cube. We also utilized the input framework provided from homework 2 for familiar usage.

2.2 Wave Particles

Wave particles are the core concept of this project and are the only 'objects' of influence that can change the scene. This meant that the surface of our fluid was unchanging unless wave particles are generated onto it. We implemented this by making a mesh surface using points like a cloth draped on a horizontal surface. This surface would then be influenced by the wave particles we generated from interactions with the surface.

Wave particles, described by Yuksel et al., are particles used to represent bumps or dents in the surface, otherwise known as a wave. These particles, represented as points on the surface of a fluid, have properties that allow for complete knowledge of where the particle is at any given time. These wave particles are spawned with its birth position, birth time, direction, angle of dispersion, and amplitude, which are all used to represent the wave at its position on the surface. This representation of a wave particle is then utilized as a height field over the surface of the wave. In other words, using the wave particles we find which points on the surface need to be moved and move them. The wave particles are also spawned with a set velocity, there is no acceleration and waves are dampened to a certain amplitude threshold before death. This simulates the presence of friction and gravity that slowly force the wave towards a flat surface again.

This implementation, provided by Yuksel et al., is modeled using a height field and can be represented by the equation

$$z(x,t) = z_0 + \eta_z(x,t)\,,$$

where the z_0 is the base height of the surface and $\eta_z(x,t)$ is the deviation field. The deviation field, $\eta_z(x,t)$, is then calculated by summing all local deviation to wave particle i

$$\eta_z(x,t) = \sum_i D_i(x,t)\,,$$

where $D_i(x,t)$ is the deviation of point x in respect to wave particle i at time t. This deviation can then be calculated using the distance between the point on the

surface to the wave particle and its amplitude. By extending this with a blending function for wave fronts, the equation is more realistic, as compared to multiple bumps or dents next to each other on the surface. For the sake of simplicity, Yuksel et al. use a simple cosine function for blending and the equation then becomes

$$D_i(x,t) = \frac{a_i}{2}\left(\cos\left(\frac{\pi|x - x_i(t)|}{r_i}\right) + 1\right)\Pi\left(\frac{|x - x_i(t)|}{2r_i}\right)$$

where a represents the amplitude, r represents the radius, and $x_i(t)$ represent the position at time t of wave particle i. The $\Pi(x)$denotes a rectangular function used by Yuksel et al. and can be described as a piecewise function where it returns 1 for $|x| < \frac{1}{2}, \frac{1}{2}$ for $|x| = \frac{1}{2}$, and 0 otherwise. These equations denote the impact a wave particle has on the point at position x and are used to find the total deviation made to the point.

Extending these functions even further into longitudinal waveforms, we can change alter the positions of the surface for a sharper wave. As only one dimension was calculated in the previous dimension, the longitudinal extension returns the new position in the remaining 2 dimensions for 3 dimensions. We can now find the position of the new point at time t with the longitudinal equation provided

$$x'(x,t) = x + \eta(x,t)\,,$$

where the deviation field, $\eta(x,t)$, now accounts for all 3 dimensions. In this sense we are only missing $\eta_{xy}(x,t)$, which can be represented as

$$\eta_{xy}(x,t) = \sum_i D_i^L(x,t)\,,$$

where $D_i^L(x,t)$ represents the longitudinal deviation for the point x on the surface at time t to wave particle i. This means that the horizontal position will change for the surface. This is applied because waves are not perfect curves and form sharp points when rising. This new longitudinal deviation accounts for this and applies a factor to the normal height deviation in the form of

$$D_i^L(x,t) = L_i\big(\widehat{u_i}\cdot(x - x_i)\big)D_i(x,t)\,,$$

where L_i is a vector function describing the longitudinal waveform and $\widehat{u_i}$ is the direction of wave particle i. Using these equations, we were able to calculate new positions for points on the surface that were affected by the wave particles being generated. This also allows for wave particle overlapping to create bigger waves.

The only real 'movement' in this model is performed by the wave particles on the surface. These particles are moved through constant acceleration with a dampening on amplitude each time step. To accurately simulate waves moving, wave particles have two unique actions during their lifetime: Subdivision and Reflection. These two actions are used to animate the wave moving through the scene.

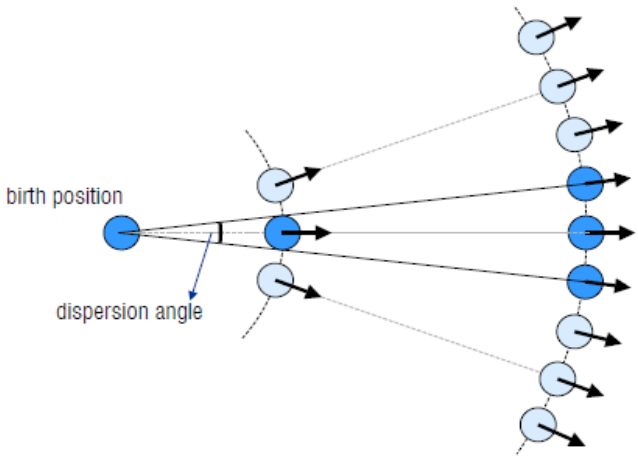

Figure 2 Subdivision

Subdivision is when the particle is more than half a radius away from its neighboring wave particles. This means that the wave front is spreading too thin and needs more particles to maintain the smooth connection between wave particles. In this process the current wave particle is subdivided into three new wave particles using the dispersion angle of the current wave particle. These new wave particles are made in the new time step to maintain smooth wave fronts because they bridge the 'gap between neighboring wave particles.

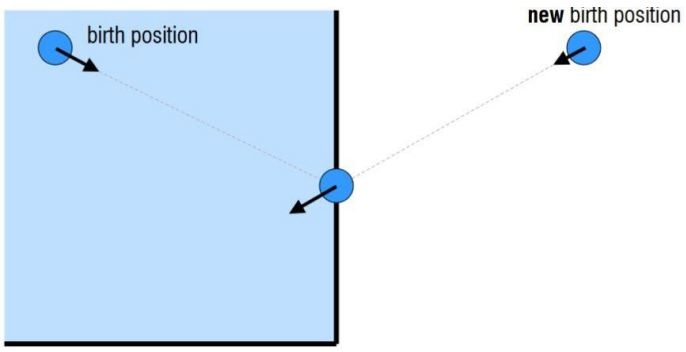

Figure 3 Reflection

Reflection, like the name implies, is the process of wave particles reflecting off surface boundaries, like walls. Reflection is handled by maintaining boundary conditions so wave particles do not extend past walls or boundaries in the scene. We had some trouble in our implementation, since checking the position at the current time did not exceed the boundary, but the next time step would. However, since the point in this case would already be outside, the new position calculated wouldn't be correct. We fixed this by checking ahead to account for this.

Utilizing these two actions for wave particles we can find the time each particle performs these actions to speed up processing time. By maintaining a constant queue of which particle needs to move at what time, we can cut down on computation for unnecessary wave particles that don't need to perform any of the two actions. This means we don't have to visit all the wave particles and can focus on only the ones that need to subdivide or reflect.

Lastly, wave particles are removed once the amplitude has lowered past a certain threshold. This allows for natural waves to die out after spawning and maintains realism in the simulation.

2.3 Wave Generation

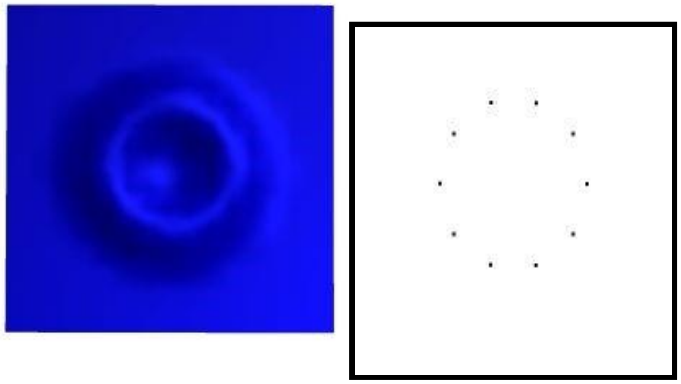

Figure 4 Wave Ring

Waves are generated when wave particles are created at points on the surface. These points are determined by the interactions made to the surface as seen in Figure 1. Yuksel et al. provides examples of wind which can generate waves at random points along the surface. An important note is that wave particles are generated in groups since a single point of interaction on a surface is near impossible in reality. This means that the surface of an object colliding with the surface of the fluid generates points along the face of the surface touching the water. In other words, the collision between water and object spawn wave particles along the faces that collide. Figure 2 shows the method of reflection as described by Wave Particles [Yuksel 2007].

For the case of wind, the surfaces interacting are arbitrary and can be generated at random. This results in differing lengths of wave fronts at random points on the surface of fluids for waves generated by the wind. Solids interacting with the surface also spawn wave particles, but in specific points around the object face colliding with the surface. These wave particles are then propagated through the scene to simulate the waves caused from the object.

3 Results

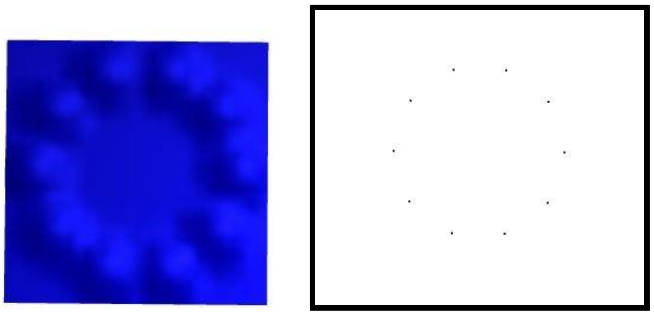

Figure 5 Failed Dispersion

The project we made to implement surface interactions of fluids used the aforementioned wave particles only. Unfortunately we were unable to finish full implementation of subdivision which leads to discontinuity over the wave fronts simulated. Our model and input are closely related to the cloth input model for homework 2, where we define the area of the surface and extended the input to include the objects that might fall onto the surface.

The generation of objects works and is randomly placed across a height specified by the input file. We did not have time to include collision detection for the objects, so spawning more than one may result in objects clipping into each other. When given a base surface and no objects, our simulation runs a simple wave produced by the wind. The wave is defined over the surface and moves in the positive x direction. The wave does imitate reality and smoothly runs and reflects over the surface and boundaries. We were unable to attach a video to this paper, but our simulation was indeed very similar to Yuksel et al. simulation of a confined wave. Subdivision was tested by spawning a ring of wave particles, which would be like simulating a drop into the surface. Since we did not have enough time to implement objects falling into the surface, we specifically defined the ring and tested for subdivision this way. This test clearly shows our lack of subdivision of wave particles and can be seen in Figure 3.

4 Conclusion

This project was motivated by our interests in simulating fluids realistically. Even though the SPH model [Ihmsen, 2014] was an alluring and interesting method, we decided to start with simple fluid surfaces and planned to extend this towards a full three part system that can include wave particles [O'Brien and Hodgins, 1995]. Due to time constraints and other projects and assignments, our time spent on this project was shorter than we would have liked. We were unable to model depth and object motion, but were able to simulate the waves themselves. In total we spent an equal amount of time working towards our current progress with a total of around 46 hours between the both of us. The results we obtained were very successful and implied the possibility for future work in extended this into a three part system.

Future work could be made into extending and fixing the current program. In regards to the objects, animation can be added as well as collision detection. The depth of the fluid can also be added by using a vertex shader for subsurface coloring. There are also extensions that we did not implement for wave particles, namely subsurface objects and their interactions with the surface.

Overall, this project showed us the possibilities we could achieve in simulating realistic fluids. In all fairness, we were not able to fully finish this project, but the potential and future definitely seem promising with the realistic results we obtained.

Acknowledgements We would like to extend our sincere gratitude to Professor Cutler for allowing us this opportunity and providing us a framework to start with. We would also like to thank Yuksel et al. in their contributions to computer graphics and fluid simulations.

References

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models Image Process.* 58, 5, 471–483.

O'BRIEN, J. F., AND HODGINS, J. K. 1995. Dynamic simulation of splashing fluids. In *CA '95: Proc. of the Computer Animation*, 198

YUKSEL, C., HOUSE, H. D., AND K, J. 2007. Wave Particles. In *ACM SIGGRAPH 2007*, 99.

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., TESCHNER, M. 2014. SPH Fluids in Computer Graphics. In *EUROGRAPHICS 2014 - State of the Art Reports*, 21-42