Simulation of Fluid Motion in a Shallow Context in 3-Dimensions

Jacob Dominy

June 16, 2010

Abstract

As computer graphics become more advanced and realistic, it becomes necessary to learn how to recreate real-life events in a virtual environment. The events that have proved most problematic in this regard are those that occur in nature. Fluids, like water and air have been especially difficult because of the number of situations and environments it is found in and the vast amount of rules that govern its behavior. In this project I will investigate techniques to automate simple, shallow fluid motion found in everyday life, and apply them to computer modeling concepts.

Keywords: Computational Fluid Dynamics, Computer Graphics, OpenGL, Navier-Stokes, Saint Venant

1 Introduction

2 Background

Computer graphics have found many uses nowadays. It's used for animated movies, video games, and simulation software. As technology has progressed over the last few years, however, the quality of this graphic design has been increased dramatically. Photorealistic animations are now becoming commonplace. However, it is no longern good enough to merely look realistic, computer generated models are expected to act realistic as well. Recreation of the motion of fluids has proved to be an enduring conundrum for graphic designers. This is because all computer graphics are based on the combined use of many 3-Dimensional objects. This makes modeling solids very simple, but very difficult when it comes to fluids. A deep understanding of physics is also required to recreate the motion of fluids as they are dictated by a large set of rules in nature. Because there are countless different situations and conditions that occur in nature, encompassing all of these possiblities has made it hard to code realistic fluid motion. Several different methods to approaching this problem have been researched in this field. The Navier-Stokes equations, first proposed in 1822 provide a way of understanding motion in incompressible fluids. The Saint Venant equations are also used, and these are based on the Navier-Stokes equations but applied in a context that does not compensate for depth. There is also an even simpler approach, called the Shallow Water equations, that were be used in this project.

2.1 Objectives

There has been a great deal of research into the field of fluid dynamics and the rules that govern fluids. The objective is this project was to determine how these rules can be applied in a simulated environment. Because this is a very broad field of study, this research focused only on liquid fluids in small, standing, and shallow contexts. This way we were able to isolate only a few variables to be concerned with. In doing this, we could ignore the factors of flow and currents, like those found in rivers and oceans, making the scope of the project more specified and easier to control.

3 Goal

The goal of this project was to create a program that can model the motion of a small, shallow body of water when disturbed, such as ripples and waves, in three dimensions.

It was meant to allow for user input to dictate the nature and starting place of the motion, as well as for the changing of certain variables that can change how the fluid operates. The program was meant to allow for the view of the fluid to be changed, letting the user rotate the model to view it from any angle and zoom in and out.

3.1 Design Criteria

There were several elements that I worked to incorporate into the project. The first was obviously be the window displaying the current 3 Dimensional model. I also planned to incorporate mouse and keyboard inputs to controls things such as the view and orientation of the model and the zooming in and out of it. In addition, I had planed to use several menus to control key variables in the project. The program was also meant to include realistic light effects and the refraction of that light was planned.

3.2 Expected Results

The goal of this project was a very ambitious one, so while I had high hopes for this project, I was uncertain of it's feasibility. My measure of success was if I can have a running simulation program that accepts all appropriate inputs and shows a basic system of using these inputs in the changing of its modeling of fluids. I definitely needed all the time I can get for working on this project to complete it. If a student next year would like to continue in the same direction as I, I would suggest the modeling of currents such as those found in rivers and oceans.

3.3 Related Work

The field of fluid dynamics is not limited to only liquids, and others have performed research regarding the flow of air and other gases [3]. This sort of project has a number of uses in areas such as aerodynamics and wind tunnel simulations. Similar methods can also be applied to the study of waves inside the earth.

4 Problem to Solve

4.1 Fluid Representation

Before even thinking about any fluid dynamics equations, we first had to consider how to represent the fluid in a computer generated environment. In the case of liquids, one of the most popular options is to use a particle system, in which each particle represents one water molecule and acts independently of the others. This is the most realistic looking method. Another, similar method is to create a large 3-Dimensional grid, called a voxel field, and have each cell act as a small section of the body of water. These methods are very intensive on computer resources, as they require calculations for a very large number of different objects. These will not be used in this project, as they are far too complicated for our needs. Instead we will use what is called a hight field. A height field is a 2-Dimensional matrix of points that represent points on the surface of our liquid body. This field is displayed on the XZ-plane to the viewer, and each point in the grid has a height value, which is then reflected on the Y-axis. This method was the most reasonable for our purposes, as our shallow body of water was not concerned with the flow of water below the surface.

4.2 Computational Fluid Dynamics

After deciding on how to represent liquid in 3-D space, we then had to figure out how to make it move. The most common method is the use of the Navier-Stokes equations, discovered in the 19th century. These equations take into account variables such as pressure, gravity, viscosity, and density to describe velocity vectors for every point in a body of liquid. This equation is applied in many different ways depending on the context and often combined with other mathematical ideas. Because our situation only concerned a shallow body of water and thus we only cared about its motion on the surface, the Navier-Stokes equation can be simplified to disregard the motion underneath the surface of the body of water. However, the Navier-Stokes equations were still too complex for the purpose of this project, so we used versions of the Shallow Water equations. Our program used them and apply it to every point in the height matrix which in turn resolved that point's height and display it accordingly. Other methods include using the Saint-Venant equations, which are modified versions of the shallow water equations, and work in a very similar way to the Navier-Stokes equations.

4.3 Assumptions

To properly derive an equation that we can properly translate into code for our program, there are several assumptions that we had to make first. The first was that the the only thing used is a height map to represent the surface of the water. This means that although we did not concern ourselves with any water underneath the surface, that also means that the motion on the surface is severely limited. It cannot splash; for that, we would've had to use a particle system. The second assumption was that we could ignore the vertical velocity of the water points. The third and final assumption was that any given point represents a column of water under it, and that the horizontal velocity at any point is constant through that column.

4.4 Discussion

In order to properly implement the equation into the program, after we have made our assumptions and integrating and discretizing the Shallow Water equations, we are left with a form of them as shown:

$$\frac{\partial u}{\partial t} + g\frac{\partial s}{\partial x} = 0$$

$$\frac{\partial s}{\partial t} + \frac{\partial}{\partial x}(ud) = 0$$

Figure 1: Version of the Shallow Water equations using Newton's Second Law and the Law of Conservation of Mass[5].

$$\frac{\partial^2 s}{\partial t^2} = -g\left[\frac{d_{x-1} + d_x}{2(\Delta x)^2}\right](s_x - s_{x-1}) + g\left[\frac{d_x + d_{x+1}}{2(\Delta x)^2}\right](s_{x+1} - s_x)$$

Figure 2: After calculations and integration, we are left with a single equation that solves for the vertical acceleration of a single point on the height field[5]. Uses the formula for velocity of a shallow water wave, which is the square root of G times depth.

$$\mathbf{A}h_i(n) = 2h_i(n-1) - h_i(n-2)$$

Figure 3: This is the final equation that will be used to calculate the height. It uses the matrix below and the heights of the two previous points to calculate height at any point.[4]

$$\mathbf{A} = \begin{pmatrix} e_0 & f_0 & & & & & \\ f_0 & e_1 & f_1 & & & & \\ & f_1 & e_2 & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & e_{n-3} & f_{n-3} & \\ & & & & f_{n-3} & e_{n-2} & f_{n-2} \\ & & & & & f_{n-2} & e_{n-1} \end{pmatrix}$$

Figure 4: Matrix A. Uses modified versions of the simplified Shallow Water equations above.[4]

However, the simplified version of the Shallow Water equations has one great flaw: it does not take into account the bottom of the water versus the height. This means that it is possible for the height to be below the floor of the body of fluid, causing the depth to be negative. This also means that the equation does not conserve volume correctly. This means that I need to check that that doesn't happen manually. With a simple method, if the volume of the body doesn't stay the same at any time, the extra positive or negative volume will be distributed across the body. This way, it will always stay the same mass. I also need to, at the beginning of the loop, create the Matrix A, and use that and the 3rd equation shown to solve for all of the heights. This is the correct method, which I was not using at the beginning. Since switching to this method, I found I could totally disregard acceleration and velocity, meaning I would not need to store them in the array of every point along with the heights. However, I needed to store two different arrays of heights, one being the current and one being the previous, because to solve for the heights, we needed the heights of each point at the two previous timesteps. This entire process is only to create a wave in one direction. To create motion in two directions, we must merely repeat this process for the X and Z axes, iterating over every row and column calculating the new heights.

5 Results

In the final version of the project, the main goal was achieved while some of the less important plans fell by the wayside. The program is able to display and update the height field in 3 dimensions, and every point's height is able to be changed. Using the mouse, the user is able to zoom, pan, and rotate the model through the implementation of the GLTZPR library. The project is split into two programs, one that creates one directional waves and another that creates two directional waves. Both programs automatically propagate waves, and do not allow for the user to create new ones. The waves that are created look very close to real fluids, however, due to the fact that the the methods that conserve volume were not implemented correctly, the fluid slowly loses volume until all of the points end up lower than they started. Despite this, the model works fairly well and serves its purpose of looking good while maintaning a level of realistic motion.

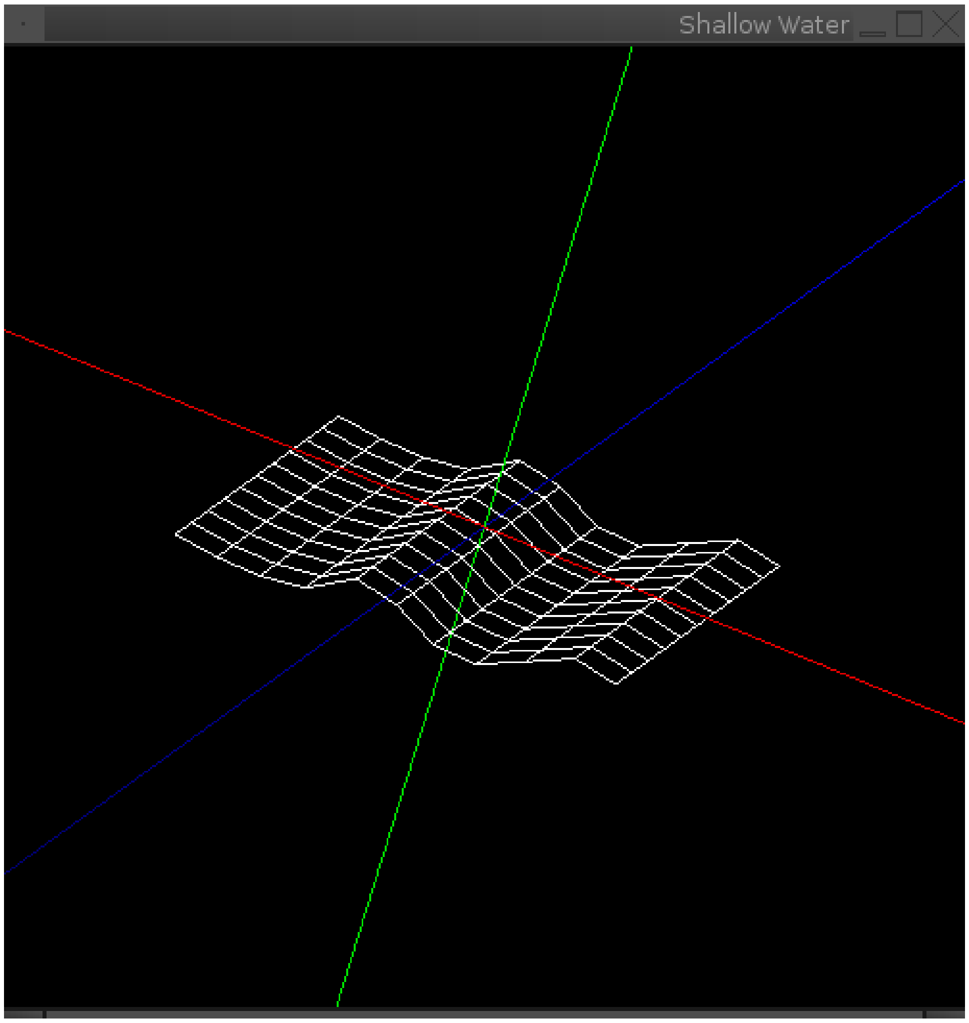

Figure 5: Unidirectional wave created by the final version of the project.

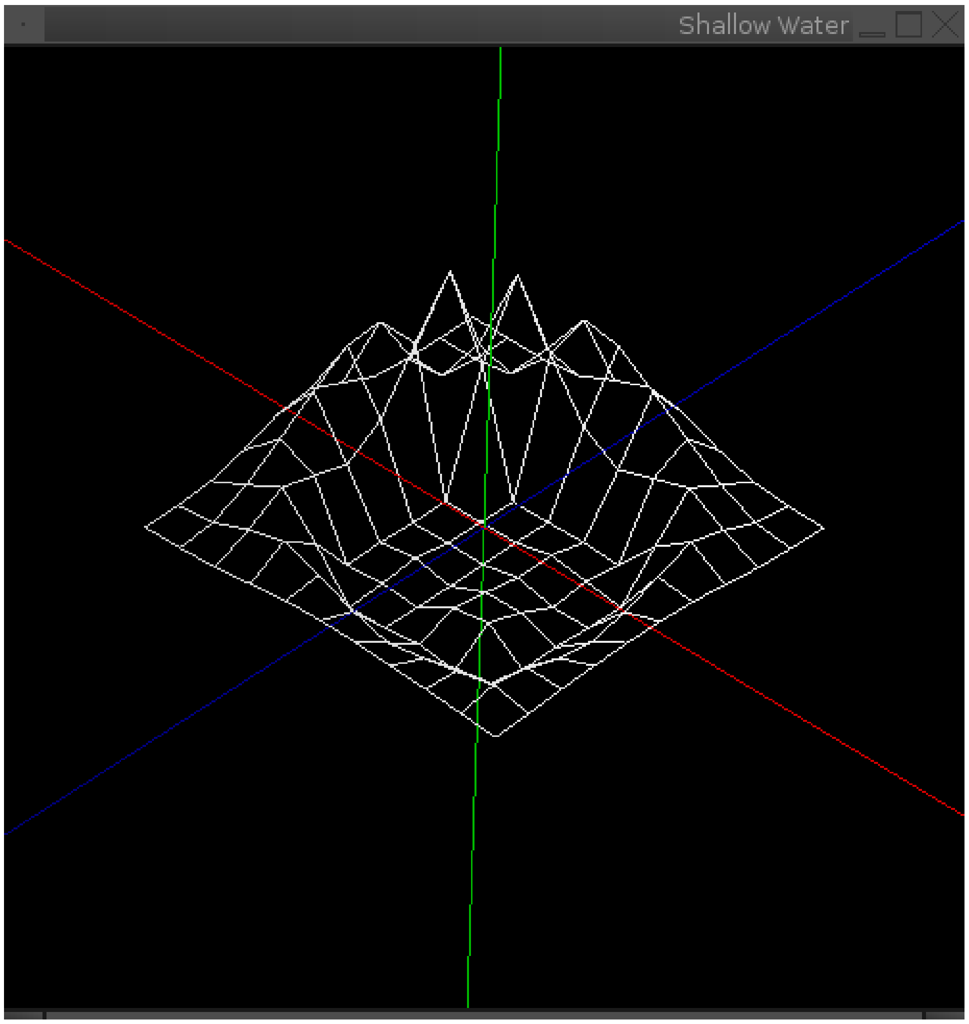

Figure 6: Bidirectional wave created by the final version of the project.

References

[1] N. Foster and R. Fedkiw, "Practical Animation of Liquids", Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 23-30, August 2001.

[2] Janghee Kim, Deukhyun Cha, Byungjoon Chang, Bonki Koo, Insung Ihm, "Practical animation of turbulent splashing water", Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, September 02-04, 2006, Vienna, Austria.

[3] Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, James F. O'Brien, "Fluid animation with dynamic meshes", ACM SIGGRAPH 2006 Papers, July 30-August 03, 2006, Boston, Massachusetts.

[4] Michael Kass, Gavin Miller, "Rapid, stable fluid dynamics for computer graphics", Proceedings of the 17th annual conference on Computer graphics and interactive techniques, pp.49-57, September 1990, Dallas, TX, USA.

[5] Jeff Lander, "Research on the Rhine: Reflections on Water Simulation", Game Developer, pp. 1-4, January 2000.

[6] G.X. Wu, Q.A. Ma, R. Eatock Taylor, "Numerical simulation of sloshing waves in a 3D tank based on a finite element method", Appl. Ocean Res. 20, pp. 337-355, 1998.

[7] Joe Stam, "Stable Fluids", ,pp. 121-128, 1999.

[8] Carcione, Jose M. and Poletto, Flavio and Gei, Davide, "3-D Wave simulation in anelastic media using the Kelvin-Voigt constitutive equation", J. Comput. Phys., pp. 282-297, 2004.

[9] Magnus Wrenninge and Doug Roble, "Fluid simulation interaction techniques, Proceedings of the SIGGRAPH 2003 conference on Sketches & applications: in conjunction with the 30th annual conference on Computer graphics and interactive techniques, July 27-31, 2003, San Diego, California.