Water Animation With Disturbance Model

Qianhua Chen Jiansong Deng Falai Chen
Department of Mathematics
University of Science and Technology of China
Hefei, Anhui 230026, People's Republic of China
Email: qhchen@263.net

Abstract

This paper provides a physically based model to animate water. A disturbance model is proposed to simulate various kinds of waves. We use a powerful solver called Finite Volume Method to solve the water fluid equation and give various kinds of disturbance to the solutions according to different disturbance sources such as wind and rain droplets. In this way, we can nicely simulate the movement of waves such as superposition and reflection, and thus easily simulate scenes of raining pool, windy lake,etc.

Keywords: computer animation, water wave, finite volume method, disturbance model

1 Introduction

Computer animation techniques have gotten adequate development in recent years. One of the main techniques is based on key frames. We can create some key frames of the scene and the frames between the key frames are produced by interpolation. But now many researchers have shown more interest in physical models to animate natural scenes. In this approach, each frame is automatically created by solving some physical equation, and the scene is totally determined by the physical rules. What we need to do is to build such a physical model and provide some parameters to control the animation.

Making water animation is an interesting problem. Many researchers have achieved remarkable results in this area. Most of the methods to animate water can be classified into four categories. The first one is based on wave synthesis. It synthesizes the wave properties and tries to use mathematical functions such as sin, cos, to simulate the shapes of water. In [1, 2], authors take the surface $y = f(x, z, t)$, where $f(x, z, t) = \sum_{i=0}^{n} A_i \cdot \sin(\omega_i t + \phi_i)$, as the water surface and modify the parameters A_i, ω_i, ϕ_i to model water movement. These approaches concern how to adjust the parameters to simulate the wave phenomena such as superposition, refraction and reflection. The second method is based on physical models, which use a fluid dynamic equation such as a 2D or 3D Navier-Stokes Equation to describe water flow. The water animation is produced by rendering the solutions of the equation at each time step. For example, in [3, 4, 5, 7], authors are concerned with building a reasonable physical model and how to solve the equations which describe the model. To make the scene more realistic, they usually add some particles to simulate sprays and foams. The third approach is based on particle systems. Researchers [8, 9] take water fluid as a volume of particles and set some rules to the particles, and then render the volume of particles. Finally, some papers focus on the rendering of a water scene. For example, [10, 11] study the interaction of light with water to produce high quality still images of water.

In this paper, we propose a new method to animate water. Our method is also based on a physical model. Our new idea is to design an effective disturbance model to simulate water scenes.

We take the water surface as a height field. We solve the two dimensional Navier-Stokes equation using a stable and fast algorithm called Finite Volume Method. Then we draw the resulting surface at each time step.In order to simulate various water scenes, we give a set of disturbances to modify the shape of the water surface. In this way we can nicely simulate the scenes of rainy pool, windy pool, and ripples.

Comparing with previous work, our work has the following features:

1. We propose a new idea to simulate the water flow. Instead of concerning ourselves with how to give an appropriate initial value of the dynamic equation, we focus on how to give a set of disturbances for the dynamic equation. We design some typical disturbances to simulate the wind blowing, stone throwing, and stick stirring. Thus we can define a sequence of disturbances to design the scenario of water animation.

2. We introduce a new solver called Finite Volume Method (FVM) to solve the dynamic equation. The solver is quite stable and fast. And we can adjust viscosity to simulate different kinds of fluids.

3. We can deal with boundary problems of the domain which has complex topology. In previous work, solvers usually use Partial Difference Method which is only suitable for a rectangular domain. With the power of FVM, we can easily simulate the scene of a pool with a tiny island in the center. And we can simulate the scene of dam breaking.

This paper is organized as follows. Section 2 discusses our point of view on the generation of water waves. Section 3 introduces a powerful solver FVM to solve the Navier-Stokes equation for water fluid. Section 4 discusses the initial conditions and boundary conditions for the fluid dynamic equation. Section 5 proposes a disturbance model to simulate various scenes of water. Section 6 describes the detailed design of the disturbance of rain droplet. Section 7 implements the algorithm. In Section 8, we discuss how to tune the viscosity of fluid, and finally in Section 9 and 10, we make some discussions and point out future work.

2 Mechanism of Wave Generation

Before we show the detailed algorithm for the water animation, we would like to make two concepts clear: water wave and water fluid. Water fluid is the body of water, which flows and has a flow velocity. Water wave is the wave which takes water fluid as a carrier. If the river bed is very regular, i.e, flat, smooth, and the bank is absolutely straight, the water fluid flows calmly and the surface of the water is just a plane. However, if we throw a stone into the river, then waves are generated and spread away. That is to say, disturbance is the cause of wave generation. The wave has a velocity that is different from flow velocity. This fact guides our new idea of designing an appropriate disturbance model to simulate interesting scenes of water.

3 Navier-Stokes Equation and Its Solver

We use the two dimensional Navier-Stokes equation to model shallow water waves. Just as stated in [3], the model is based on three approximations. First, the water surface is taken as a height field in a three dimensional space. Second, the vertical component of the velocity of the water particle is ignored. Thirdly, the velocity of the water in a vertical column are approximately constant. The limitation of these approximations makes the model be only suitable for the cases where the main body of the water doesn't break. However, the experience of hydrodynamicists and the wide usage

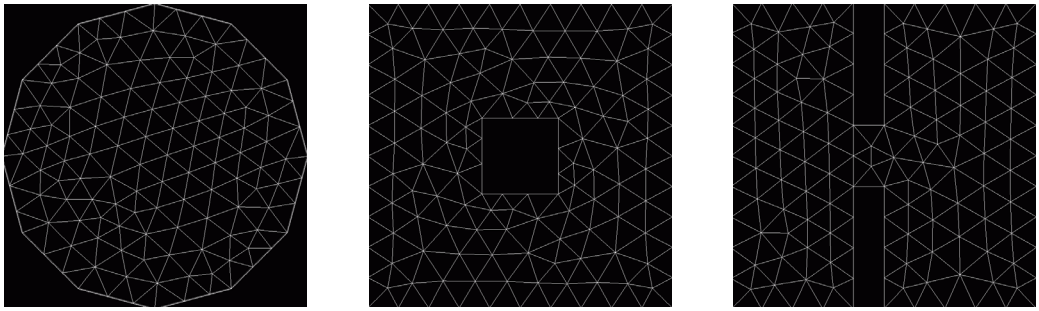

Figure 1: Discretizing the domain into meshes

of this model in numerical analysis show that it is an effective model to simulate water scenes in a wide range.

Let h be the depth of the water and (u, v) be the planar components of the velocity of the water in a vertical column. The shallow water equation can be written in conservative form as follows [15]:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = 0$$

or

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{U} d\Omega + \int_{\partial \Omega} \mathbf{F} \cdot \mathbf{n} ds = 0 \tag{1}$$

where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$, $\mathbf{F} = (\mathbf{E}, \mathbf{G})$. The vector of unknowns,i.e., the height and velocity of water body, is

$$\mathbf{U} = \begin{pmatrix} h \\ uh \\ vh \end{pmatrix}$$

And the Cartesian components of the flux are

$$\mathbf{E} = \begin{pmatrix} uh \\ u^2 h + \frac{1}{2} g h^2 \\ uvh \end{pmatrix}, \mathbf{G} = \begin{pmatrix} vh \\ uvh \\ v^2 h + \frac{1}{2} g h^2 \end{pmatrix}, \mathbf{n} = \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

where Ω is the solution domain, g is the acceleration due to gravity, and $\mathbf{n}$ is the normal vector of the domain boundary.

3.1 Discretization

The domain is discretized into a set of triangular cells. We use the freeware EasyMesh [12] to do this work. The domain which has complex topology can also be discretized into triangles (Figure 1). Our aim is to get $\mathbf{U} = (h, hu, hv)$ for each cell and use the height h of each cell to produce the whole water surface.

3.2 Finite Volume Method

Finite Volume Method is an efficient method to solve PDEs. Since equation (1) is the governing equation for all the water body, each discretized cell of the domain should also obey it. Let A_e be

the directional area of the triangular cell, $\mathbf{U}_e$ be the value of $\mathbf{U}$ on this cell, $l_j, j = 1, 2, 3$ are the length of the three walls of the cell (Figure 2), we have

$$A_e \cdot \frac{\partial \mathbf{U}_e}{\partial t} + \sum_{j=1}^{3} (\mathbf{F} \cdot \mathbf{n})_j \cdot l_j = 0 \tag{2}$$

for each cell.

Now compute the following values:

1. $\dfrac{\partial \mathbf{U}_e}{\partial t}$. We use the simple differential scheme to compute:

$$\frac{\partial \mathbf{U}_e}{\partial t} = \frac{\mathbf{U}_e^{n+1} - \mathbf{U}_e^n}{\Delta t}$$

2. A_e. Let the three vertices of cell e be (x_1, y_1), (x_2, y_2), (x_3, y_3), then the directional area of the cell is

$$A_e = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

3. l_j. The length of walls of the cell e is

$$l_j = \sqrt{(x_{j+1} - x_{j+2})^2 + (y_{j+1} - y_{j+2})^2}$$

where the index j should be moduled by 3.

4. $(\mathbf{F} \cdot \mathbf{n})_j$. We use the technique proposed by Roe ([?]) to compute this part. Let $\mathbf{U}_j^+$ and $\mathbf{U}_j^-$ be two states on both sides of the wall j of the cell e at the time level $n\Delta t$(figure 2), then

$$\begin{aligned} (\mathbf{F} \cdot \mathbf{n})_j &= \frac{1}{2}\big[(\mathbf{F} \cdot \mathbf{n})_j^+ + (\mathbf{F} \cdot \mathbf{n})_j^- - |\mathbf{A}|(\mathbf{U}_j^+ - \mathbf{U}_j^-)\big] \\ &= \frac{1}{2}\big[\mathbf{F}(\mathbf{U}_j^+) \cdot \mathbf{n}_j + \mathbf{F}(\mathbf{U}_j^-) \cdot \mathbf{n}_j - |\mathbf{A}|(\mathbf{U}_j^+ - \mathbf{U}_j^-)\big] \end{aligned}$$

where $\mathbf{A}$ is the Jacobian matrix of the projection of the flux $\mathbf{F}$ in the normal direction evaluated at some average of the variables at states $\mathbf{U}^+$, $\mathbf{U}^-$:

$$\mathbf{A} = \frac{\partial(\mathbf{F} \cdot \mathbf{n})}{\partial \mathbf{U}} = \begin{pmatrix} 0 & n_x & n_y \\ (c^2 - u^2)n_x - uvn_y & 2un_x - vn_y & un_y \\ -uvn_x + (c^2 - u^2)n_y & vn_x & un_x + 2vn_y \end{pmatrix}$$

and $|\mathbf{A}|$ is the matrix whose elements are absolute values of elements of $\mathbf{A}$.

The following are details of the calculation:

(a) $\mathbf{n}_j$. The normal vectors of the walls are

$$\mathbf{n}_j = \frac{1}{l_j}\big((y_{j+2} - y_{j+1}), -(x_{j+2} - x_{j+1})\big).$$

Figure 2: Calculate $\mathbf{U}_j^+$, $\mathbf{U}_j^-$ for each cell

(b) $|\mathbf{A}|(\mathbf{U}_j^+ - \mathbf{U}_j^-)$. We need more computation for this part [15]. The solution space of $|\mathbf{A}|(\mathbf{U}_j^+ - \mathbf{U}_j^-)$ can be represented by the eigenvectors of matrix $|\mathbf{A}|$, that is:

$$|\mathbf{A}|(\mathbf{U}_j^+ - \mathbf{U}_j^-) = \sum_{k=1}^{3} \alpha_k \cdot |\lambda_k| \cdot \mathbf{e}_k$$

where $|\lambda_k|, k = 1, 2, 3$, are the absolute values of the eigenvalues of matrix $|\mathbf{A}|$ and $\mathbf{e}_k$ are the corresponding eigenvectors.

$$\begin{aligned} \lambda_1 &= un_x + vn_y + c \\ \lambda_2 &= un_x + vn_y \\ \lambda_3 &= un_x + vn_y - c \end{aligned}$$

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ u + cn_x \\ v + cn_y \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ -cn_y \\ cn_x \end{pmatrix}, \mathbf{e}_3 = \begin{pmatrix} 1 \\ u - cn_x \\ v - cn_y \end{pmatrix}, c = \sqrt{gh}$$

and α_k are coefficients of the decomposition in the basis of eigenvectors of λ_k,

$$\begin{aligned} \alpha_1 &= \frac{1}{2}\Delta h + \frac{1}{2c_\alpha}\left[\Delta(uh)n_x + \Delta(vh)n_y - (u_\alpha n_x + v_\alpha n_y)\Delta h\right] \\ \alpha_2 &= \frac{1}{c_\alpha}\left[(\Delta(vh) - v_\alpha \Delta h)n_x - (\Delta(uh) - u_\alpha \Delta h)n_y\right] \\ \alpha_3 &= \frac{1}{2}\Delta h - \frac{1}{2c_\alpha}\left[\Delta(uh)n_x + \Delta(vh)n_y - (u_\alpha n_x + v_\alpha n_y)\Delta h\right] \end{aligned}$$

where

$$u_\alpha = \frac{u^+\sqrt{h^+} + u^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}}, \quad v_\alpha = \frac{v^+\sqrt{h^+} + v^-\sqrt{h^-}}{\sqrt{h^+} + \sqrt{h^-}}, \quad c_\alpha = \sqrt{g\frac{h^+ + h^-}{2}}$$

and

$$\begin{aligned} \Delta h &= h^+ - h^- \\ \Delta(uh) &= (uh)^+ - (uh)^- \\ \Delta(vh) &= (vh)^+ - (vh)^- \end{aligned}$$

(c) $\mathbf{U}_j^+$, $\mathbf{U}_j^-$. For simplicity, we use the scheme with the accuracy of order 1. That is, we take the value of $\mathbf{U}$ at the center point of neighboring cell as $\mathbf{U}_j^+$, and the value of $\mathbf{U}$ at the center point of the current cell as $\mathbf{U}_j^-$ (figure 2).

$$\mathbf{U}_j^+ = \mathbf{U}_f, \quad \mathbf{U}_j^- = \mathbf{U}_e$$

Now if we have the value of $\mathbf{U}_e$ at time level $n\Delta t$, from step 1 we can get the value at time level $(n+1)\Delta t$:

$$\mathbf{U}_e^{n+1} = \mathbf{U}_e^n - \frac{\Delta t}{A_e}\sum_{j=1}^{3}(\mathbf{F}\cdot\mathbf{n})_j \cdot l_j$$

We compute the unknown parts by the steps listed above and get the solution $\mathbf{U}_e^{n+1} = (h, uh, vh)_e^{n+1}$, we produce the surface made up of all the triangles with vertices (x_i, y_i, h_i), $i = 1, 2, 3$ (each triangle corresponds to each cell), and take it as the water surface.

3.3 Numerical Stability

The stability of the FVM scheme can be judged by the following formula:

$$\Delta t \leqslant \frac{\min\{d\mathbf{r}_{i,j}\}}{2\max\{(c + \sqrt{[u^2 + v^2]_{i,j}})\}} \tag{3}$$

where $d\mathbf{r_{i,j}}$ represents the whole set of distances between every center point (i, j) and those of its adjacent cells. In simple words, let A be the minimum of all the cell areas, the condition can be described as

$$\frac{\Delta t}{A} \leqslant \text{a certain value.}$$

4 Initial and Boundary Conditions

To start the time evolution computation on the two-dimensional domain, we have to specify the values of h, u, v at every center point of each cell for time $t = 0$, and assuming the values are uniform over each cell. Unlike previous works [4], we don't have to design complicated initial values for the animation scene, we just give the cells some trivial values. For example, to simulate the water in a pool, we specify all the cells the same values $(h_0, 0, 0)$. To simulate the scene of dam breaking, we set the initial values to be $(h_1, 0, 0)$ for the cells in one part of the domain and the initial values to be $(h_2, 0, 0)$ for the cells in the other part of the domain.

We set the boundary conditions by calculating $\mathbf{U}_j^+$, $\mathbf{U}_j^-$ for each boundary wall of the boundary cells. We set two kinds of boundary conditions for the equation (2). One is wall reflection condition (close boundary condition). By this condition setting, we can easily simulate scenes where waves meet with the bank of a river or pool and are reflected. We calculate the values $\mathbf{U}_j^+$, $\mathbf{U}_j^-$ at the boundary wall of cell e as follows[Figure 2 and 3]:

$$\mathbf{U}_j^+ = \overline{\mathbf{U}}_e = \begin{pmatrix} \overline{h} \\ \overline{hu} \\ \overline{hv} \end{pmatrix}, \mathbf{U}_j^- = \mathbf{U}_e = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}$$

here, $(\overline{h}, \overline{hu}, \overline{hv})$ is the result of $\mathbf{U}_e = (h, hu, hv)$ being reflected at the boundary wall of the cell e.

The other boundary condition is an outflow condition (open boundary condition). This is for simulating the scene where water flows from a pool into a river. The waves aren't reflected at the border of the domain, but go into another domain. When calculating the values at the boundary wall of cell e, we take the values at the center point of the cell as $\mathbf{U}_j^+$, $\mathbf{U}_j^-$. That is:

$$\mathbf{U}_j^+ = \mathbf{U}_e = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \mathbf{U}_j^- = \mathbf{U}_e = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}$$

5 Disturbance Model

The water body is governed by the Navier-Stokes equation, and the initial values are trivial. Now how can we simulate various water scenes? Just like a painter making a drawing on a blank canvas,

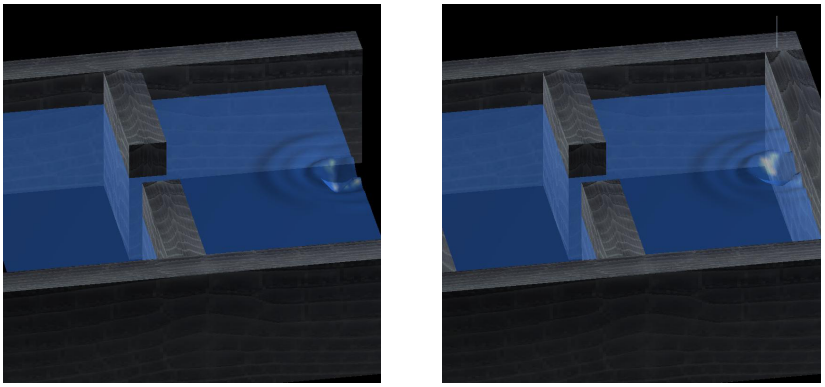

Figure 3: Open boundary and close boundary

we add different kinds of disturbances to the water and generate various shapes of waves and water scenes. At each time step of solving Navier-Stokes equation, we disturb the solution by

$$\mathbf{U}^n += d\mathbf{U}$$

i.e., we modify $\mathbf{U}^n$ on some cells by $h+ = dh, u+ = du, v+ = dv$, and we take the disturbed $\mathbf{U}^n$ as the new initial values for $\mathbf{U}^{n+1}$. Since the water is governed by the Navier-Stokes equation, the neighboring cells are influenced by the disturbance and their heights and velocities are changed. Thus waves are generated and spread out automatically. When more than one waves are generated, waves can meet and superpose with each other. When the waves meet the boundary of the domain, they are reflected. When the waves meet the break of the boundary, they go into another water domain. In this way, the wave actions such as reflection, superposition are simulated automatically and perfectly.

We take the rain droplets, wind, creatures in the water and boats, etc. as disturbance sources. Each disturbance source has its own way to influence the water surface by disturbing a different set of cells and providing different amounts of disturbances to their heights and velocities.

To simulate a disturbance, two main factors have to be considered. One is the action of the disturbance source, and the other is the influence when the source meets with the water surface. In this paper, we only provide the detailed design for the disturbance of rain droplet. We will enrich the model to include more kinds of disturbances in the future work.

6 Simulation of Rain Droplets

We first consider the action of a rain droplet. It is guided by the forces of gravity $\mathbf{G}$, air friction $\mathbf{f}$, and wind blowing $\mathbf{F}$ (figure 4). So the movement of the droplet can be described as

$$\begin{aligned} \mathbf{F} + \mathbf{G} + \mathbf{f} &= m\mathbf{a} \\ \mathbf{v} &= \mathbf{v_0} + \mathbf{a}\Delta t \\ \mathbf{p} &= \mathbf{p_0} + \mathbf{v}\Delta t \end{aligned}$$

where Δt is length of time step, $\mathbf{v}$, $\mathbf{p}$ are current velocity and position respectively, $\mathbf{v_0}$, $\mathbf{p_0}$ are velocity and position respectively at the previous time step. The mass m is determined by the size of droplet and friction $\mathbf{f}$ is determined by the velocity.

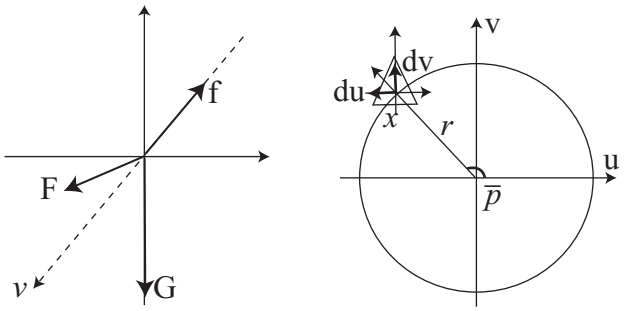

Figure 4: Movement and influence of droplet

Now considering the influence of a rain droplet when it meets with the water surface. It disturbs a circular area of water. The radius of the circle is determined by the droplet's size and speed. The cells inside the circle will be disturbed by dh, du, dv, which are also determined by the size and the speed of the droplet. That is (Figure 4),

$$\mathbf{r} = \mathbf{x} - \overline{\mathbf{p}},$$
$$||\mathbf{r}|| \leq factor\_radius \cdot droplet\_size \cdot droplet\_speed$$
$$dh = factor\_dh \cdot droplet\_size \cdot droplet\_speed$$
$$du = factor\_du \cdot droplet\_size \cdot droplet\_speed$$
$$dv = factor\_dv \cdot droplet\_size \cdot droplet\_speed$$

Where $\mathbf{r}$ is the vector from the center of considered cell $\mathbf{x}$ to the droplet's projection on water plane $\overline{\mathbf{p}}$. The speed of the droplet can be calculated by $droplet\_speed = ||\mathbf{v}||$. And $factor\_dh$, $factor\_du$, $factor\_dv$ and $factor\_radius$ are coefficients.

The whole disturbing procedure can be described as follows: a droplet is falling to the water and we check each cell of the water surface to see which cell the droplet is going to meet with. If the droplet's current position and its next position cross a certain cell, we say the droplet meets with the cell and causes disturbance in the way we have described above. Then the droplet is bounced or merged into water.

7 Implementation of Algorithm

We describe the pseudo code of our algorithm as follows:

```
For each time step n do
{
    //1. calculate U+, U- for each cell
    calculate U_j+, U_j- for the walls of each boundary cell, deal with boundary conditions;
    calculate U_j+, U_j- for the walls of each internal cell;
    //2. add disturbance to invoke waves
    for each disturbance source
    {
        calculate the disturbance source's current position and velocity;
        check each cell to see if the source has influence on it, and disturb it if there exists
        influence;
    }
```

//3. $\mathbf{U}^n \rightarrow \mathbf{U}^{n+1}$

for each cell calculate $\mathbf{U}_e^{n+1} = \mathbf{U}_e^n - \frac{\Delta t}{A_e} \sum_{j=1}^{3} (\mathbf{F} \cdot \mathbf{n})_j \cdot l_j$*;*

//4. $\mathbf{U}^n \leftarrow \mathbf{U}^{n+1}$*, reset* $\mathbf{U}^n$ *to be* $\mathbf{U}^{n+1}$*, and take them as new initial values for the next //time step.*

for each cell reset $\mathbf{U}^n$ *to be* $\mathbf{U}^{n+1}$*;*

//5. produce the water surface

for each cell calculate its normal and its three vertices (x_i, y_i, h_i) *and then rendering;*

}

8 Viscosity of Fluid

There is an interesting point worthy to mention. As we know from section 3, the stability of solution of the N-S equation is determined by $\Delta t/A$. If $\Delta t/A$ exceeds a certain value, the solution will be unstable, otherwise the solution is stable. In fact, $\Delta t/A$ indicates the viscosity of fluid. The larger $\Delta t/A$ is, the less viscous the fluid looks. So we can tune $\Delta t/A$ to simulate many kinds of fluids which have different viscosity.

However, when a mesh is created (and A is determined), sometimes even you tune Δt to its maximum value, you may still feel the viscosity of the water is too large. Here we provide a smart trick to "enlarge" Δt. That is, we only render the water surface every other time step, then we achieve the effect of doubling Δt.

9 Demo

We apply our algorithm to simulate the scene of light rain. In the demo attached in this paper, a pool is created and some rain droplets are added. The pool is calm at first. Then it rains and the water surface is disturbed. Thus the ripples are generated and spread away. Note that the waves are reflected when they meet with the bank of the pool and the ripples are superposed with each other nicely.

We illustrate the movement of water waves in Figure 5. The pictures captured from the animation show that the phenomena of water waves can be produced automatically. When waves meet with each other, they are superposed. When waves meet the bank of the pool, they are reflected. If there is a column block in the water, waves can go around it and continue propagating. All these movements, i.e., superposition, reflection and rounding, are generated automatically.

Our application is programmed using C++ and OpenGL on an SGI Octane workstation. If we discretize the domain into about 932 triangular cells, the animation is in real time, with the speed of 17 frames a second. (Note that the scene has been mapped with textures). However, if we discretize the domain into 10,192 cells, as in the demo, our program can only produce two frames per second. We need to point out that the speed of solving the Navier-Stokes equation is quite fast. Without rendering the pictures, the speed is about 43 frames a second for 932 cells.

10 Conclusions and Future Work

The demo shows that the movement of waves is realistic. This is an effective way to simulate the water scene in which the main water body doesn't break. We are confident that we have found a good way to simulate the generation of water waves. This work is a success of integration of computer graphics and numerical analysis. However, there is still much work to do. One piece of future work is to enrich the disturbance model to simulate wind blowing, stick stirring, boat rowing, etc. While writing this paper, we are simulating wind blowing. Another piece of future

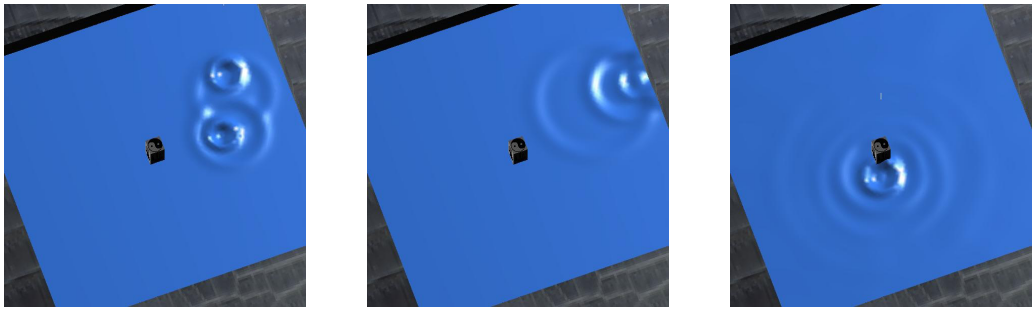

Figure 5: Superposition, reflection and rounding of waves

work is to make shadows and add particle systems to make the water scene more realistic. Finally, how about the cases where the river bed or pool bottom is not plane? This is also a challenging problem.

Acknowledgement

Prof. Liu Ruxun and Mr. Wang Jiwen gave us great help in solving the Navier-Stokes equation. This work is supported by the National Natural Science Foundation of China (19971087) and the Research Fund for the Doctoral Program of Higher Educational Committee.

References

[1] Darwyn R. Peachey, Modeling Waves and Surf. SIGGRAPH '1986, 65-74.

[2] Alain Fournier, A Simple Model of Ocean Waves. SIGGRAPH '1986, 75-84.

[3] Michael Kass, Gavin Miller, Rapid, Stable Fluid Dynamics for Computer Graphics. SIGGRAPH '1990, 49-55.

[4] Ying Qing Xu, Cheng Su, etc, Physically Based Simulation of Water Currents and Waves, Chinese Journal of Computers, 1996, volume 19 (supplement), 153-160.

[5] James F. O'Brien, Jessica K. Hodgins,Dynamic Simulation of Splashing Fluids. Proceedings of Computer Animation '1995, 198-205.

[6] Jos Stam, Stable Fluids, SIGGRAPH '1999, 121-128.

[7] Karl Sims, Particle Animation and Rendering Using Data Parallel Computation. SIGGRAPH '1990, 405-413.

[8] Gavin Miller, Andrew Pearce. Globular Dynamics: a Connected Particle System for Animating Viscous Fluids. Computers & Graphics, 1989, volume 13, 305-309.

[9] Mark Watt,Light-Water Interaction using Backward Beam Tracing. SIGGRAPH '1990, 377-385.

[10] Simon Premoze, Michael Ashikhmin, Rendering Natural Waters. Proceedings of Pacific Graphics '2000, 23-30.

[11] Bojan Niceno, http://www-dinma.univ.trieste.it/ nirftc/research/easymesh/easymesh.html

[12] Francisco Alcrudo, Pilar Garcia-Navarro, A High-Resolution Godunov-Type Scheme in Finite Volumes For The 2D Shallow-Water Equations, International Journal For Numerical Methods in Fluids, 1993, volume 16, 489-505.

[13] N. Foster, D. Metaxas. Modeling Water for Computer Animation. Communications of the ACM. July 2000, 43(7), 60-67.