Stable But Nondissipative Water

OH-YOUNG SONG, HYUNCHEOL SHIN, and HYEONG-SEOK KO
Seoul National University

This article presents a physically-based technique for simulating water. This work is motivated by the "stable fluids" method, developed by Stam [1999], to handle gaseous fluids. We extend this technique to water, which calls for the development of methods for modeling multiphase fluids and suppressing dissipation. We construct a multiphase fluid formulation by combining the Navier–Stokes equations with the level set method. By adopting constrained interpolation profile (CIP)-based advection, we reduce the numerical dissipation and diffusion significantly. We further reduce the dissipation by converting potentially dissipative cells into droplets or bubbles that undergo Lagrangian motion. Due to the multiphase formulation, the proposed method properly simulates the interaction of water with surrounding air, instead of simulating water in a void space. Moreover, the introduction of the nondissipative technique means that, in contrast to previous methods, the simulated water does not unnecessarily lose mass, and its motion is not damped to an unphysical extent. Experiments showed that the proposed method is stable and runs fast. It is demonstrated that two-dimensional simulation runs in real-time.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Animation*

General Terms: Algorithm

Additional Key Words and Phrases: CIP method, multiphase fluid, natural phenomena, Navier–Stokes equation, physically based animation, semi-Lagrangian method, stable fluids, water

1. INTRODUCTION

Water, which covers two thirds of the earth, undergoes myriad types of motion in its constant interactions with air, solids, and living creatures. Water has featured prominently in several recent feature animations, including *Finding Nemo* and *The Perfect Storm*. The success of these movies depended greatly on visual effects in the animation of water. Physically-based approaches have been shown to effectively reproduce water movement, with quite impressive results [Foster and Fedkiw 2001; Enright et al. 2002].

However, several open challenges remain in this field. One key issue is speeding up the simulation of water. In the case of gaseous phenomena, interactive simulation methods already have been introduced by Stam [1999]. The method is called "stable fluids", which allows a large simulation time step to be used without causing instabilities. Unfortunately, this method is known to suffer from large amounts of numerical dissipation, which results in loss of mass. This is not important when simulating dissipative media such as fog or smoke, but it is not tolerable when animating intrinsically

This research was supported by the Korea Ministry of Information and Communication. This research was also partially supported by the Automation and Systems Research Institute at Seoul National University and the Brain Korea 21 Project.
Authors' address: Graphics and Media Lab, School of Electrical Engineering #001, Seoul National University, San 56-1, Shillim-dong, Kwanak-ku, Seoul, 151–741, Korea; email: song@graphics.snu.ac.kr.

nondissipative substances like water. Another undesirable property of the stable fluids method that must be noted is numerical diffusion, which dampens the fluid motion. Although damping is an inherent property of all fluids, the damping caused by numerical diffusion in the stable fluids method is too severe. Therefore, if we wish to simulate water using an approach based on the stable fluids method, we must modify that method to prevent the numerical dissipation and reduce the numerical diffusion.

This article presents a new physically-based method for simulating water. The proposed method, which is based on the semi-Lagrangian methodology, retains the speed and stability of the stable fluids technique, while including mechanisms to fix the problems of numerical dissipation and diffusion. To obtain nondissipative water, we adopt the constrained interpolation profile (CIP) method, which has been shown to remarkably reduce dissipation due to the use of coarse grids. To prevent dissipation due to the use of a large time step, we propose a novel particle-based approach, which we show to be quite effective at preventing dissipation of small-scale features. This particle-based approach is also used to simulate droplets and bubbles, which contributes to the overall visual realism. In addition, compared to existing methods, the proposed method simulates water-air interactions more accurately by employing the multiphase dynamic equations that account for the presence of air.

The rest of the article is organized as follows: Section 2 reviews previous work; Section 3 formulates the multiphase fluid; Section 4 describes the CIP-based fluid solver; Sections 5 and 6 present our particle-based technique for preventing dissipation; Section 7 reports our experimental results; and finally, Section 8 concludes the article.

2. PREVIOUS WORK

Early work on physically-based simulation of water for graphics applications concentrated on animating the height-field representation of the water surface. To obtain interactive performance, researchers used the two-dimensional (2D) approximation of the Navier–Stokes equations. Kass and Miller [1990] generated the height fields using an approximate version of the 2D shallow water equations. To simulate water–object interactions, Chen and Lobo [1995] solved the 2D Navier–Stokes equation that includes pressure. O'Brien and Hodgins [1995] proposed a method for simulating splashing liquids by integrating a particle system into a 2D height-field model.

Height fields cannot be used to represent water that is undergoing a highly dynamic motion such as pouring. To handle such motions, researchers turned to the 3D Navier–Stokes equations. Foster and Metaxas [1996, 1997a] animated 3D liquids by modifying the Marker and Cell method proposed by Harlow and Welch [1965]. In addition, Foster and Metaxas [1997b] simulated gases by using an explicit finite difference approximation of the Navier–Stokes equations. Stam [1999] introduced the unconditionally stable fluid model, which utilizes the semi-Lagrangian method in combination with an implicit solver. This model gave significantly improved simulation speeds, but suffered from numerical dissipation. To reduce the dissipation in simulations of gaseous fluids, Fedkiw et al. [2001] proposed the use of vorticity confinement and cubic interpolation. Based on the stable semi-Lagrangian frame-work, Rasmussen et al. [2003] proposed an efficient method for depicting large-scale gaseous phenomena, and Feldman et al. [2003] proposed an explosion model that incorporated a particle-based combustion model into the semi-Lagrangian framework. Treuille et al. [2003] proposed a constrained optimization technique for keyframe control of fluid simulations and McNamara et al. [2004] improved the optimization speed drastically by adapting the adjoint method.

In order to handle 3D liquids, the semi-Lagrangian scheme must be augmented with a robust and accurate method for tracking the liquid surface. To address this issue, Foster and Fedkiw [2001] proposed a novel method for representing a dynamically evolving liquid surface, which was based on combining the level set method with massless marker particles. Enright et al. [2002] improved this hybrid scheme by introducing the "particle level set method", which could capture water surface with a remarkable

accuracy. The particle level set method was employed in recent studies of fluids by Carlson et al. [2004], Goktekin et al. [2004], and Losasso et al. [2004].

Takahashi et al. [2003] simulated multiphase fluids by employing the CIP method, coupled with the volume of fluid scheme; their method simulated the water-air interaction properly, instead of simulating water in a void space. When we are to animate water at an interactive rate, as demonstrated by Stam [1999] in the case of gas, then the use of large time steps should be allowed. But it can cause dissipation of mass. In Foster and Fedkiw [2001] and Enright et al. [2002], the time step size had to be restricted to prevent loss of mass. Although the CIP scheme used by Takahashi et al. [2003] lessened the degree of the dissipation, loss of mass was still noticeable when large time steps were used.

Several particle-based methods have been proposed as alternatives to the above grid-based approaches. Miller and Pearce [1989] simulated fluid behavior using particles connected with viscous springs. Terzopoulos et al. [1989] adopted a molecular dynamics model to simulate particles in the liquid phase. Stam and Fiume [1995] introduced "smoothed particle hydrodynamics" (SPH) to depict fire and gaseous phenomena. In SPH, the fluid is modeled as a collection of particles with a smoothed potential field. Premože et al. [2003] introduced the use of the moving particle semi-implicit method (MPS) for simulating incompressible multiphase fluids. One drawback of particle-based methods is that, if insufficient particles are used, they tend to produce grainy surfaces. To prevent this, a sufficiently large number of particles must be used, which increases the computational cost.

3. FORMULATION AND OVERVIEW

In order to produce realistic movement of water in the presence of air, we base our method on the multiphase Navier–Stokes equations in combination with the level set method. The multiphase Navier–Stokes equations can simultaneously represent both water and air. The level set method, which can represent the water-air interface as an implicit surface, has been shown to be a robust method for capturing topological changes of water surfaces. Furthermore, the surface curvature can be accurately calculated from the level set values, and hence the surface tension, which is proportional to the curvature, can be easily incorporated into the dynamic simulation.

We start by introducing the incompressible Navier–Stokes equations for a multiphase fluid. Let $\mathbf{u} = (u, v, w)$ denote the velocity field of the fluid. Then, the flow of fluid is described by

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

and

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \frac{\mathbf{f}}{\rho} + \frac{\nu}{\rho}\nabla^2 \mathbf{u} - \frac{\nabla p}{\rho}, \tag{2}$$

where p is the pressure, ρ is the density, ν is the kinematic viscosity, and $\mathbf{f}$ represents the external forces per volume. Equations (1) and (2) state that mass and momentum, respectively, should be conserved. To treat the immiscible multiphase fluid consisting of water and air within a single formulation, we employ the level set function ϕ [Osher and Sethian 1988; Sussman et al. 1994]. ϕ is an implicit signed distance function defined to be positive for water, and negative for air. Thus, the sign of ϕ also determines the density and viscosity of the medium.

The water dynamically evolves in space and time according to the underlying fluid velocity field $\mathbf{u}$. The updates in the level set values due to $\mathbf{u}$ are expressed by the level set equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0. \tag{3}$$

The surface of water, which will be a focal point throughout this work, can be obtained by tracking the locations for which $\phi = 0$.

To solve the above equations numerically, we divide the space into a finite number of cells.We evaluate the pressure and level set values at the cell center, but we evaluate the velocity at the center of each cell face. This approach is the classical staggered grid discretization [Harlow and Welch 1965], which naturally enforces boundary condition and avoids the checkerboard pattern of the pressure [Trottenberg et al. 2001]. At each time step, our simulator performs the following three steps:

(1) *Advect the level set:* The level set ϕ is advected according to Equation (3), which causes the density and viscosity fields appearing in Equation (2) to be updated.

(2) *Update the velocity:* Equation (2) is solved for $\mathbf{u}$ using the following procedure [Stam 1999]: (1) calculate the advection component $\mathbf{u} \cdot \nabla \mathbf{u}$ using the semi-Lagrangian method; (2) apply the forces $\mathbf{f}/\rho$; (3) add the effect of the viscous term $\nu/\rho\nabla^2\mathbf{u}$ by employing implicit central differencing; and (4) project the velocity field so that the condition $\nabla \cdot \mathbf{u} = 0$ is met.

(3) *Simulate droplets/bubbles:* Droplets/bubbles are identified and simulated using the particle dynamics until they merge into the body of water/air.

Execution of the above procedure produces the ϕ and $\mathbf{u}$ of the next time step. The method for implementing Steps 1 and 2 is presented in the following section, and the implementation of Step 3 is described in Section 5.

4. CIP-BASED FLUID SIMULATOR

The framework of our fluid simulator is based on the semi-Lagrangian scheme, which was briefly introduced in Section 3. A detailed description of the semi-Lagrangian scheme can be found in Stam [1999] and Staniforth and Côtè [1991]. In the method proposed here, we make several modifications to the previous semi-Lagrangian scheme to reduce the numerical dissipation and diffusion. This section describes those modifications.

4.1 CIP Advection

In the semi-Lagrangian scheme, advection is implemented by referring to the function value at $\mathbf{x}-\mathbf{u}\Delta t$.[1] Since the physical values of ϕ and $\mathbf{u}$ are defined only at discrete points, the function values for $\mathbf{x} - \mathbf{u}\Delta t$ can be obtained by linearly interpolating the neighboring grid points. This approach is computationally efficient and unconditionally stable [Stam 1999]. However, it may smooth out the subcell features. This problem, referred to as nonphysical numerical diffusion, causes the movement of fluid to be excessively damped, which hampers the generation of turbulent effects such as the formation of water droplets or air bubbles in violently interacting multiphase fluids. It also causes dissipation of the mass.

Fortunately, an anti-diffusive technique called the Constrained Interpolation Profile (CIP) method is proposed by Yabe and Aoki [1991] and Yabe et al. [2001]. We adopt the CIP method when solving Equation (3), resulting in a reduction of mass dissipation, and also when solving the advection term $\mathbf{u} \cdot \nabla \mathbf{u}$ in Equation (2), leading to a reduction in the degree of damping. Adoption of the CIP method allows us to simulate phenomena such as turbulent flows or swirls with reasonable visual quality.

The key idea of the CIP method is to use not only the function values at the grid points, but also the spatial derivatives at those points for constructing the profile inside the grid cell. For example, in the one-dimensional case, the profile corresponding to $[x_i, x_{i+1}]$ now has four conditions ϕ_i, ϕ_i', ϕ_{i+1}, and

[1] In fact, more sophisticated backtracking methods can be used. In this work, we used the second order Runge-Kutta backtracking.

ϕ'_{i+1}, and can be represented by the third order polynomial

$$\Phi(X) = [(aX + b)X + \phi'_i)]X + \phi_i, \tag{4}$$

where $X = x - x_i$ for $x \in [x_i, x_{i+1}]$. The coefficients a and b can be expressed in terms of the four conditions:

$$a = (\phi'_i + \phi'_{i+1})/\Delta x^2 - 2\Delta\phi/\Delta x^3,$$
$$b = 3\Delta\phi/\Delta x^2 - (2\phi'_i + \phi'_{i+1})/\Delta x,$$

where $\Delta x = x_{i+1} - x_i$ and $\Delta\phi = \phi_{i+1} - \phi_i$. The spatial derivatives used in the CIP method are directly determined from the differentiated version of the original advection equation.

To advect the level set values, we differentiate Equation (3) with respect to ξ, which gives

$$\frac{\partial\phi_\xi}{\partial t} + \mathbf{u}\cdot\nabla\phi_\xi = -\mathbf{u}_\xi\cdot\nabla\phi, \tag{5}$$

where $\phi_\xi = \partial\phi/\partial\xi$, $\mathbf{u}_\xi = \partial\mathbf{u}/\partial\xi$, and ξ is one of the spatial variables (x, y, z). The task of solving the above equation for ϕ_ξ can be performed in two steps: first, solving the nonadvective part $\partial\phi_\xi/\partial t = -\mathbf{u}_\xi\cdot\nabla\phi$ using finite differencing; then advecting the result according to

$$\frac{\partial\phi_\xi}{\partial t} + \mathbf{u}\cdot\nabla\phi_\xi = 0. \tag{6}$$

Noting that the advections of Equations (3) and (6) are driven by the same velocity field $\mathbf{u}$, we can advect both ϕ and (ϕ_x, ϕ_y, ϕ_z) by referring to the same point of the profile. For the one-dimensional case, suppose that the grid point x_j is backtracked to $x_r \in [x_i, x_{i+1}]$. Then, the advected results ϕ_j and ϕ'_j can be obtained by evaluating Equation (4) and its differentiated form at $X_r = x_r - x_i$. Namely, $\phi_j = \Phi(X_r)$ and $\phi'_j = \Phi'(X_r)$.

A problem can arise if we use the profile of Equation (4) as it stands. In the one-dimensional case, for example, the value $\Phi(X_r)$ may lie outside the range $[\phi_i, \phi_{i+1}]$, which can cause instabilities. One solution to this problem is the rational CIP method proposed by Xiao et al. [1996], which suppresses the above oscillation problem by using a rational function instead of a polynomial function. In the present work, we developed an alternative CIP scheme that explicitly modifies the derivatives, with this modification being applied only to the cells in which oscillations are present. This scheme, which is described in detail in the Appendix, guarantees a monotonic profile; hence, we call it monotonic CIP. Because it uses polynomials, monotonic CIP runs faster than rational CIP.

Higher (i.e., two- or three-) dimensional CIPs have been proposed by Yabe et al. [2001]. In those methods, however, the derivative constraints are not applied at every grid point, which can result in a nonmonotonic profile. Here we implement higher dimensional CIPs based on the one-dimensional monotonic CIP solver. Our implementation of higher dimensional CIPs, which is described below, is always stable. Consider the two-dimensional case shown in Figure 1(a), where $\mathbf{r}$ is the backtracked point. For this system, we must determine $[\phi, \phi_x, \phi_y, \phi_{xy}]_\mathbf{r}$ from the values at the four corners: (i, j), $(i+1, j)$, $(i, j+1)$, and $(i+1, j+1)$. Here, $[\cdot]_\mathbf{x}$ denotes the function and derivative values at point $\mathbf{x}$. The one-dimensional solver can determine $[\phi, \phi_x]_\mathbf{a}$ from $[\phi, \phi_x]_{(i,j)}$ and $[\phi, \phi_x]_{(i+1,j)}$, and $[\phi_y, \phi_{xy}]_\mathbf{a}$ from $[\phi_y, \phi_{xy}]_{(i,j)}$ and $[\phi_y, \phi_{xy}]_{(i+1,j)}$. Similarly, it can determine $[\phi, \phi_x, \phi_y, \phi_{xy}]_\mathbf{b}$. Then, it can determine $[\phi, \phi_y]_\mathbf{r}$ from $[\phi, \phi_y]_\mathbf{a}$ and $[\phi, \phi_y]_\mathbf{b}$, and $[\phi_x, \phi_{xy}]_\mathbf{r}$ from $[\phi_x, \phi_{xy}]_\mathbf{a}$ and $[\phi_x, \phi_{xy}]_\mathbf{b}$.

In the three-dimensional case, the CIP implementation additionally calls for the values of ϕ_{xy}, ϕ_{yz}, ϕ_{xz}, and ϕ_{xyz}. Obtaining those values by analytic differentiation of the original formula involves a large amount of computation. However, our experiments indicated that approximating the second order and

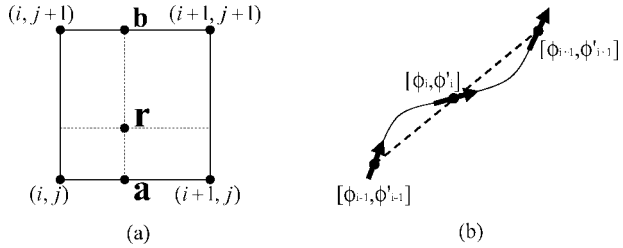

Fig. 1. (a) Two-dimensional CIP interpolation; the x-axis is along the horizontal direction and the y-axis is along the vertical direction. (b) Schematic of a situation where ϕ_{i-1}, ϕ_i, and ϕ_{i+1} are aligned.

higher derivatives by finite differencing of the first order derivatives caused no visually noticeable difference in the result, but significantly reduced the amount of computation.

We close this section by noting the attractive aspects of CIP in comparison with other interpolation methods. In cases where ϕ_{i-1}, ϕ_i, and ϕ_{i+1} are aligned, such as that shown in Figure 1(b), spline techniques that do not utilize the derivative information interpret ϕ as being straight. In contrast, because CIP utilizes the spatial derivatives ϕ'_{i-1}, ϕ'_i, and ϕ'_{i+1} of the original equation, it results in more accurate modeling of the real situation. Therefore, the CIP method allows us to use a fairly coarse grid. Another advantageous feature of the CIP method is that, whereas high-order interpolation methods for the semi-Lagrangian scheme such as the cubic spline, quintic Lagrange [Staniforth and Côtè 1991], and monotonic cubic polynomial [Fedkiw et al. 2001] methods require three stencils (or four grid points) to construct the profile in the one-dimensional case, the profile used in the CIP method can be constructed with information from a single cell. This feature is particularly useful when treating boundaries.

4.2 Force

We now apply the external forces, **f**, to the result of the advection. The external forces consist of gravity and surface tension. In the interactive system we develop, mouse or keyboard input can also be used to generate additional forces. The gravitational force is expressed as $\rho\mathbf{g}$, where **g** is the gravitational acceleration. The surface tension is given by

$$\mathbf{f}_{st} = -\rho\sigma\kappa(\phi)\delta_\varepsilon(\phi)\nabla\phi, \tag{7}$$

where σ is a constant coefficient, κ is the local curvature given by

$$\kappa(\phi) = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{8}$$

δ_ε is a smeared delta function which, in the present work, has the form

$$\delta_\varepsilon(\phi) = \begin{cases} \frac{1}{2\varepsilon} + \frac{1}{2\varepsilon}\cos\left(\frac{\pi\phi}{\varepsilon}\right) & : \quad |\phi| \le \varepsilon, \\ 0 & : \quad otherwise, \end{cases} \tag{9}$$

where we use $\varepsilon = 1.5\Delta x$ for the smearing width. Equation (7) is the continuum surface force (CSF) model proposed by Brackbill et al. [1992]. It allows us to treat surface tension as a body force, and does not require explicit information on the geometry and position of the water surface.

In order to show surface tension effects accurately, a strict time step restriction of $O(\Delta x^{1.5})$ is required [Brackbill et al. 1992; Sussman et al. 1994]. Since we do not restrict time steps in this work, the effects

may be somewhat suppressed. In large scale liquid phenomena such as the ones experimented in this article (Section 7), however, the surface tension effects are visually not significant.

4.3 Projection

The result obtained by applying the external forces then goes through the diffusion step, $\nu/\rho\nabla^2\mathbf{u}$, for which we use the implicit solver following the approach described in Stam [1999]. To process the last term of Equation (2), $-\nabla p/\rho$, we impose the fluid incompressibility condition represented in Equation (1), which produces the Poisson equation:

$$\nabla \cdot \left(\frac{\nabla p}{\rho}\right) = \frac{\nabla \cdot \tilde{\mathbf{u}}}{\Delta t}, \tag{10}$$

where $\tilde{\mathbf{u}}$ is the intermediate velocity field obtained by processing Equation (2) up to the diffusion term. Equation (10) can be discretized as

$$\begin{aligned} \sum_{n=\{i,j,k\}} \left(\rho^{-1}_{n+\frac{1}{2}} + \rho^{-1}_{n-\frac{1}{2}}\right) p_n - \rho^{-1}_{n+\frac{1}{2}} p_{n+1} - \rho^{-1}_{n-\frac{1}{2}} p_{n-1} \\ = -\tfrac{1}{\Delta t} \sum_{n=\{i,j,k\}} \left(\tilde{u}_{n+\frac{1}{2}} - \tilde{u}_{n-\frac{1}{2}}\right), \end{aligned} \tag{11}$$

where $p_{n\pm1}$ are the pressure values taken from the centers of the neighboring cells, and $\tilde{u}_{n\pm\frac{1}{2}}$ and $\rho_{n\pm\frac{1}{2}}$ are the velocity and density values taken from the cell faces. We can assemble the collection of equations of the form of Equation (11), covering the domain space into a large linear system

$$\mathbf{A}\mathbf{p} = \mathbf{b}, \tag{12}$$

where $\mathbf{p}$ is the vector of unknown pressures required to make the velocity field divergence free. $\mathbf{A}$ is a positive-definite, symmetric matrix in which density is a variable in space. This system can be efficiently solved using the incomplete-Cholesky preconditioned conjugate gradient method (ICCG) [Golub and Loan 1996].[2] The solution of Equation (12) is then used to calculate ∇p at the cell faces. Finally, we obtain the divergence free velocity field by

$$\mathbf{u} = \tilde{\mathbf{u}} - \Delta t \frac{\nabla p}{\rho}. \tag{13}$$

5. PREVENTION OF DISSIPATION IN SMALL-SCALE FEATURES

Although the CIP-based simulator described in the previous section can represent the content of each cell up to the third order, it still suffers from the following two problems: (1) when a large time step is used, it produces non-negligible mass errors; and (2) it cannot represent subcell-level features such as water droplets and air bubbles. In this section, we develop a novel technique to solve these problems. The proposed technique uses particles[3] to complement the grid-based framework, thereby overcoming some of the limitations of that framework.

[2]Because we use the conjugate gradient method for solving the linear system which is the most time consuming part, the overall time complexity order in our method is approximately $O(n^{3/2})$ in the two-dimensional case and $O(n^{4/3})$ in the three-dimensional case, where n is the total number of grid points.

[3]Note that our use of particles should not be confused with that proposed by Enright et al. [2002]. They use massless marker particles to obtain a more accurate water surface, whereas we use physical particles that have mass as well as volume to represent water droplets and bubbles. Compared to their method, our method requires far fewer particles and, as a consequence, has a negligible computational overhead.

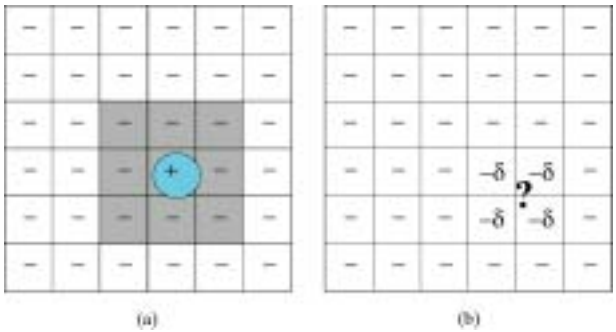

Fig. 2. Identification of droplets: (a) Apparent droplet, (b) Unapparent droplet. In (b), δ is a small positive value such that $\delta < \varepsilon$.

5.1 Identification of Dissipative Cases

If the level set value of a particular cell center is positive, whereas those of adjacent cell centers are negative, we can interpret the isolated region to be a droplet as shown in Figure 2(a). A potential problem arising from having such a small isolated region is that it may dissipate or even be lost in subsequent simulation. In such cases, therefore, we transform the region into a droplet that undergoes Lagrangian motion from that moment onwards. We determine the volume[4] of the droplet based on the level set values. The approximation we use is

$$V_f = \int_{\Omega_c} H(\phi(\mathbf{x}))d\,\mathbf{x} \approx \sum_c H_\varepsilon(\phi(\mathbf{x}_c))\Delta x \Delta y \Delta z, \tag{14}$$

where the set Ω_c represents the droplet (the circular region of Figure 2(a)), c is the index ranging over the cells that cover Ω_c (the shaded cells of Figure 2(a)), and H_ε is the smeared Heaviside function[5]

$$H_\varepsilon(\phi) = \begin{cases} 0 & : \quad \phi < -\varepsilon \\ \frac{1}{2} + \frac{\phi}{2\varepsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\phi}{\varepsilon}\right) & : \quad |\phi| \le \varepsilon \\ 1 & : \quad \phi > \varepsilon. \end{cases} \tag{15}$$

As in Equation (9), we use $\varepsilon = 1.5\Delta x$. Since the fluid content of the isolated region is already transformed into a droplet, we reduce the level set values of the region to values that are small enough (e.g., $-\varepsilon$) that the region will not be mistakenly identified as containing water during subsequent processing. Generation of bubbles can be thought of as the dual of that of droplets; identification of isolated regions and approximation of bubble volume can be done with the same procedure and same equations, except that, in this case, we use the negated version of original level set values.

In addition to the cases in which the isolated region can be identified by simply looking at the signs of the level set values, referred to here as “apparent” cases, there also exist “unapparent” cases in which the isolated region cannot be identified from the sign of the level set values. For example, consider the case in which a 2×2 grid with small negative level set values is surrounded by the cells with large negative level set values, as shown in Figure 2(b). Since the level set function we use represents signed distance, small negative values imply that water is nearby. Therefore, the situation considered here can be reconciled only by introducing a droplet somewhere around ?-marked location of Figure 2(b). The volume of the droplet is again computed by Equation (14). During dynamic simulation of a multiphase

[4]Since we assume incompressibility of both media, mass-preservation is equivalent to volume-preservation.
[5]We also smear the discontinuities in density/viscosity across the air-water interface using this function.

fluid, the above situation can occur when a body of water is splitting (or two bodies of air are merging). Cases of unapparent bubbles are treated similarly.

There is another source of errors of misinterpreting the phase, which is related to the backtracking in semi-Lagrangian advection. Specifically, when a large time step is used, backtracking may leave some cells never referred. In such cases, the masses of those cells are lost in the subsequent time steps. We prevent this type of error by converting the nonreferred cells into droplets/bubbles. Again, the volumes of the cells are determined by evaluating Equation (14) for the nonreferred cells. The droplets/bubbles are then advected by the underlying velocities. Droplets/bubbles that are advected into the water/air medium are ignored.

In pure grid-based methods, the above cases are beyond the resolution limit and thus no droplet/bubble is formed. By introducing a small amount of extra computation, however, the procedures described above can generate droplets/bubbles. This not only reduces the mass dissipation but also enhances the visual details of the dynamically evolving water.

The volume approximation of Equation (14), which uses the smeared Heaviside function given in Equation (15), theoretically has first-order accuracy. Our experiments showed that it is computationally efficient and gives sufficient visual realism for our purposes. If greater accuracy is needed, contouring methods such as the marching cube algorithm [Lorensen and Cline 1987] can be used.

5.2 Dynamic Simulation of Droplets/Bubbles

As a droplet/bubble advances with the initial velocity, it experiences gravitational and drag forces, as well as pressure from the surrounding fluid, which cause the fragment to accelerate or decelerate. The forces acting on the fragment can be summarized as

$$\mathbf{f} = m_f \mathbf{g} + \alpha_d r^2 (\overline{\mathbf{u}} - \dot{\mathbf{x}}) \|\overline{\mathbf{u}} - \dot{\mathbf{x}}\| - V_f \nabla p, \tag{16}$$

where m_f is the mass, V_f is the volume, α_d is the drag coefficient, r is the radius, $\dot{\mathbf{x}}$ is the current velocity of the fragment, and $\overline{\mathbf{u}}$ is the interpolated velocity of the grid-based fluid measured at the center of the fragment. The third term, which represents the force due to the pressure difference, produces buoyancy. In Equation (16), the second and third terms model the interaction with the neighboring fluid. Therefore, the action force given in Equation (16) must be coupled with the reaction force $-\alpha_d r^2 (\overline{\mathbf{u}} - \dot{\mathbf{x}}) \|\overline{\mathbf{u}} - \dot{\mathbf{x}}\| + V_f \nabla p$, acting on the grid-based fluid model. The movement of fragments over time is obtained by standard numerical integration. If two or more fragments overlap during the dynamic simulation, they are merged into a single larger fragment.

5.3 Restitution of Droplets/Bubbles

When either of the following two conditions are met, we restitute the droplets/bubbles to the grid-based fluid model: (1) when the volume of a fragment becomes larger than twice the cell size, or (2) when a fragment hits the surface or moves into the same phase fluid.

In the first case, the fragment has become too big to be represented as a nondeforming particle, and thus its behavior is better modeled within the grid-based framework. Therefore, in such cases, we remove the fragment and restitute its mass to the grid-based fluid model. This restitution is executed by updating the level set values and setting the grid point velocities to the fragment velocity. We update the level set values by

$$\phi(\mathbf{x}_i) = s_p (r_p - |\mathbf{x}_i - \mathbf{x}_p|), \tag{17}$$

where $s_p = +1$ for the case of a water droplet, and -1 for the case of an air bubble, r_p is the radius of the fragment, $\mathbf{x}_p$ is the center of the fragment, and $\mathbf{x}_i$ is the grid point being updated.

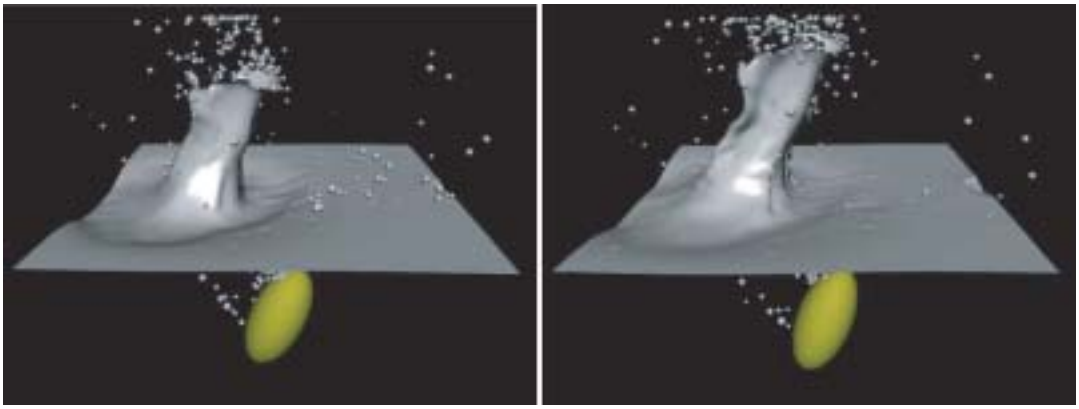

Fig. 3. Instantaneous hollows and ripples on the water surface created by restitution of Lagrangian droplets/bubbles.

The second case corresponds to the situation in which a droplet/bubble returns to the body of water/air, and, therefore, we remove the fragment. In the case where the fragment hits a surface cell, the cell velocity is updated by taking the average of the previous cell velocity and the fragment velocity. As for the level set values, we determine the new values for the cells covering the fragment by taking the inverse functions of Equations (14) and (15). In the case where the fragment moves into the same phase fluid, we perform the same procedure as described above, pretending that it hit a surface cell.

The above procedure we devised for updating the level set values and cell velocities, interestingly, contributes to creating visual details at water surface. The procedure in fact forms small ripples: a droplet falling into water contributes a downward velocity to the cell, which generates an instantaneous hollow. But soon, the region is pushed back and forms a small bump (See Figure 3.).

5.4 Variations for Visual Realism

In the above procedures, several parameters can be controlled to adjust the visual aspects of the simulation. (1) In order to produce splashes and bubbles of smaller scales, we can generate multiple droplets/bubbles of smaller size, instead of generating a single fragment of about the cell size. In practice, we let the number of generated fragments be proportional to the velocity of the fluid portion being converted. (2) We can model the geometrical shape of a fragment as an ellipsoid instead of a sphere, in accordance with the velocity. (3) In order to represent foam at the water surface, instead of removing bubbles immediately after they appear on the surface, we assign them a life time that is inversely proportional to the bubble size.

6. ADDITIONAL CONSIDERATIONS

6.1 Reinitialization of Level Set Values

As the level set is advected and makes interactions with the droplets/bubbles as described in the previous section, the signed-distances can be corrupted. This may lead to the development of noisy features that can be problematic when approximations such as finite-differencing are used. For example, such noisy features can lead to large errors in the curvature calculation. For this reason, we need to introduce a procedure to rectify the level set to ensure that it maintains the signed distance property. This procedure is called reinitialization or redistaincing [Sussman et al. 1994, 1998; Peng et al. 1999; Sethian 1996;

Tsai et al. 2003]. We used Sussman's method [1998]. The method is performed using

$$\frac{\partial \phi}{\partial \tau} + sgn(\phi)(|\nabla \phi| - 1) = 0, \tag{18}$$

where τ is a fictitious time and $sgn(\phi)$ is a smeared signum function given by

$$sgn(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla \phi|^2 (\Delta x)^2}}. \tag{19}$$

In most cases, the corruption is not severe and the procedure recovers the correct signed distance values within several fictitious time steps. Hence, the reinitialization step is computationally inexpensive. We can speed up the procedure by reinitializing only the portion within a band of the water surface [Sussman et al. 1998]. Details on the discretization of Equation (18) can be found in Sussman et al. [1998] and Osher and Fedkiw [2002].

6.2 Interaction with Rigid Objects

This section presents a simple technique for simulating the influence of water on rigid objects, as well as that of rigid objects on water. We employ the volume of solid technique proposed by Takahashi et al. [2003] with modifications inspired by the work of Foster and Fedkiw [2001] and Enright et al. [2002]. The proposed technique may not produce valid results in an engineering sense, but it does reproduce the physical aspects of object-water interactions with acceptable visual realism.

When a rigid object is immersed in water, we mark the fluid cells whose centers are contained within the object. A self-evident constraint in this case is that the fluid should not flow into the rigid object; this can be checked for each marked cell s by calculating $\mathbf{u_s} \cdot \mathbf{n_s}$, where $\mathbf{u_s}$ is the fluid velocity, and $\mathbf{n_s}$ is the normal of the object surface. If $\mathbf{u_s} \cdot \mathbf{n_s} < 0$, then the fluid is flowing into the solid and, to stop this, we remove the normal component of $\mathbf{u_s}$, while leaving the tangential component unchanged. The above procedure is for calculating the change in fluid velocity caused by the presence of an object. The movement of an object caused by the surrounding fluid is simulated by considering the fluid pressure acting on the surface of the object. Given that the gradient of the pressure generates forces, the total external force $\mathbf{F}$ and moment $\mathbf{T}$ acting on the center of mass $\mathbf{r}_c$ of the rigid object can be expressed as

$$\mathbf{F} = M\mathbf{g} + \sum_s (-\nabla p_s \cdot \mathbf{n_s})\mathbf{n_s} \Delta S, \tag{20}$$

$$\mathbf{T} = \sum_s (\mathbf{r}_s - \mathbf{r}_c) \times (-\nabla p_s \cdot \mathbf{n_s})\mathbf{n_s} \Delta S, \tag{21}$$

where M is the mass of the object, s is the index ranging over the marked cells, p_s is the fluid pressure of the cell, $\mathbf{r}_s$ is the position of the cell, and ΔS is the area of the object surface subsumed in the cell. When simulating water that is interacting with a quickly moving rigid object, a reasonably small time step size must be used to calculate the movement of rigid object in a moderate level of accuracy.

6.3 Discussion on Mass Errors

The procedures described in Sections 4 and 5 are quite effective at blocking mass dissipation in the numerical simulation of multiphase fluids. However, the errors resulting from the use of finite-sized grids and finite time steps are inevitable, and in rare cases, these errors can accumulate to some extent. In situations where the amount of water needs to be kept constant, we optionally perform a global compensation procedure based on modification of the level set values at every time step.

If the initial volume of water is V_0, and the volume at each time step is V_i, we should compensate the water volume by $\Delta V_i = V_0 - V_i$ at the end of each simulation step. The volume V_i and the area of water surface S_i can be calculated using

$$V_i = \int_\Omega H(\phi(\mathbf{x}))d\,\mathbf{x} \approx \sum H_\varepsilon(\phi(\mathbf{x}))\Delta x\Delta y\Delta z + V_p, \tag{22}$$

$$S_i = \int_\Omega \delta(\phi(\mathbf{x}))|\nabla\phi(x)|d\,\mathbf{x} \approx \sum \delta_\varepsilon(\phi(\mathbf{x}))|\nabla\phi(x)|\Delta x\Delta y\Delta z, \tag{23}$$

where V_p is the total volume of the droplets. Because the level set is a signed distance function, we can use the approximation

$$\Delta V_i \approx S_i\Delta\phi, \tag{24}$$

which allows us to calculate the scalar value $\Delta\phi$. Now, adding $\Delta\phi$ to the previous level set values of the entire domain (for both water and air) results in raising the water level by $\Delta\phi$, while not changing the geometrical shape of the water surface.

The above compensation procedure can have a first-order error, since we use the smeared Heaviside function. However, the compensation procedure performed at every time step is done with respect to V_0, and the error is distributed over all of the cells in the system. Therefore, even though the procedure is not physically-based, it effectively prevents the error accumulation without producing any noticeable artifacts.

The contribution of this procedure in the prevention of mass dissipation is small. For example, in the 2D water drop experiments on a grid of resolution 64×48, the procedure compensated only 0.7% of errors. Furthermore, contribution from the global compensation decreases as we increase the resolution of the grid. In fact, in obtaining the 3D demos, we did not run the compensation procedure.[6]

7. EXPERIMENTAL RESULTS

The technique presented in this article is implemented in two dimensional and three dimensional versions, on a PC with an Intel Pentium 4 3.2 GHz processor, 1 GB memory. The innovations introduced in the present work make possible the real-time simulation of the movement of 2D water. Moreover, the fast simulation speed of the proposed method enabled highly interactive control of the water motion through several types of action. For example, by clicking or dragging the mouse, we could add a volume of water, create/remove solid walls, or introduce force fields, all with immediate effect. This section summarizes our experimental results.

The 2D simulator based on a grid of resolution 64×48 ran at $30 \sim 60$ fps, which is sufficiently fast to enable real-time simulation of water that is subjected to interventions such as applying forces or adding water volumes. Clip 1 contains a real-time video capture taken during the simulation (see Figure 4 for a snapshot). Although the CFL number (i.e., $\mathbf{u}\Delta t/\Delta x$) was 25 or higher, the simulation ran stably. The mass error is not evident, and the movement of water is clearly nondissipative. We were able to let ink mix over the water medium, as shown in Stam [1999], which provided a perceptual cue on the movement of water. The clip shows (1) the movement of water at different gravitational conditions, (2) the generation of droplets/bubbles in turbulent water, (3) the behavior of water when interacting with solids, (4) changes in buoyancy due to changes in the density of the fluid, (5) the result of replacing the CIP advection with linear advection, which noticeably increases dissipation and damping, and (6) the interactions water makes with nonvoid air due to our multiphase implementation of the fluid.

[6]In very coarse grids (e.g., 15×15), or when very large time steps are used (e.g., the CFL number $\mathbf{u}\Delta t/\Delta x > 50$), the procedure can produce a large amount of modification in the position or curvature. In such cases, the effect of compensation can be noticeable.

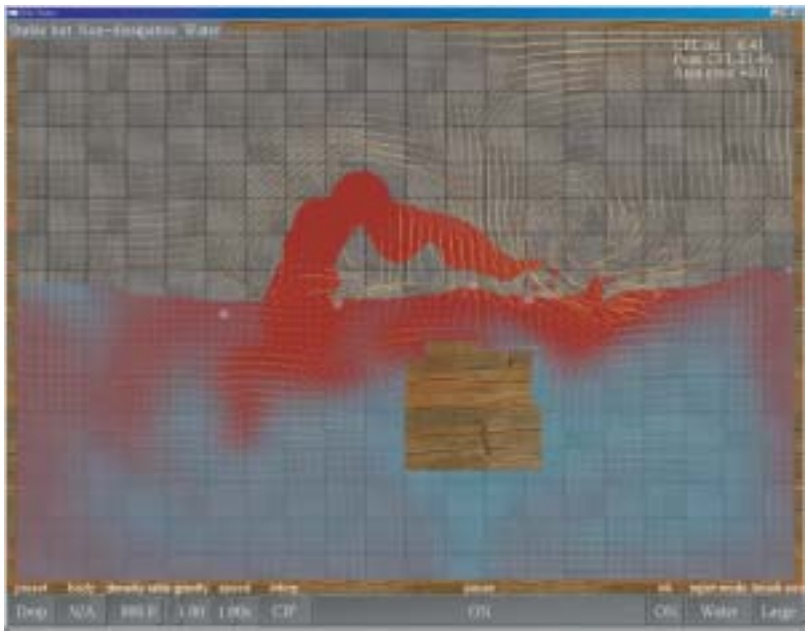

Fig. 4. Real-time 2D simulation.

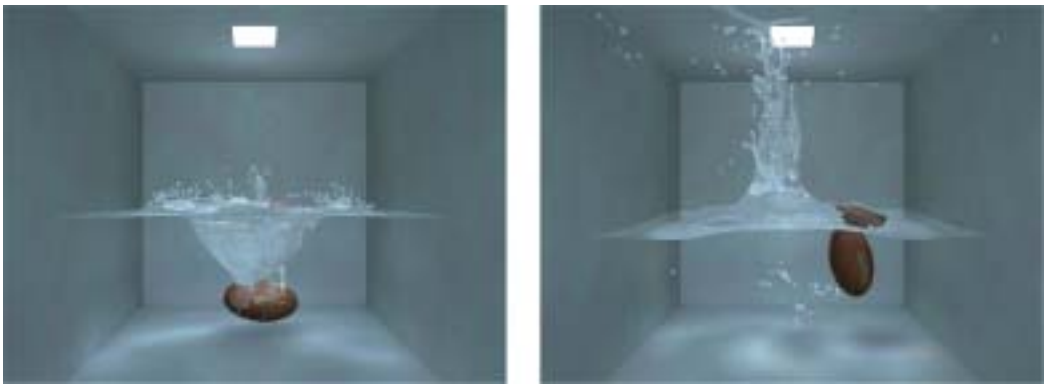

Fig. 5. A football thrown into water.

We ran the 3D version of our simulator to experiment the cases shown in Figures 5, 6, and 7, and produced the animations shown in Clips 2, 3, and 4, respectively. In the case shown in Figure 5, a football was thrown into water with a large initial velocity, which realistically produced a violent splash of water and a trail of air bubbles behind the ball. The density ratio of the ball relative to water was 0.75. In the case shown in Figure 6, a cubic container containing water and an object was rotated. The collision the water made with the wall created a highly dynamic water surface and plenty of droplets and bubbles. In the case shown in Figure 7, an empty cup was drowned into water, which caused the air to be released and to produce a dynamic response with water. If only water-phase was simulated, water would fill the cup without experiencing any resistance, thus the cup would submerge smoothly, which is not physically correct. We constrain all fluid fragments to be within the fluid simulation domain. So, the

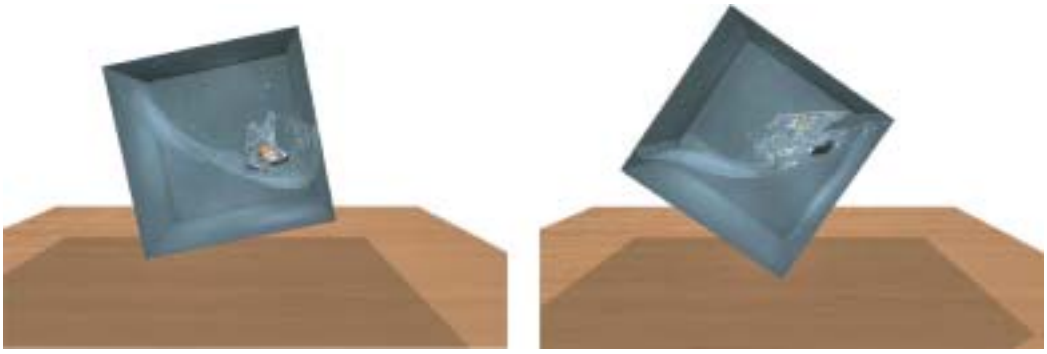

Fig. 6. Simulation of water in a rotating cubic container.

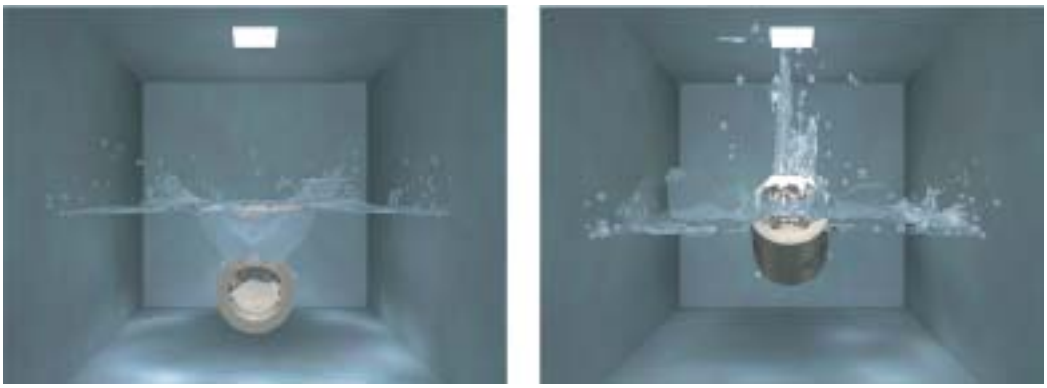

Fig. 7. A cup drowned into water.

Table I. 3D Simulation Information (Density Ratio is *Air:Water:Object*. All experiments are performed on the same condition except object densities; the viscosity coefficient of water is 1.0×10^{-2}, the viscosity coefficient of air is 2.0×10^{-4}, and the surface tension coefficient is 0.5.)

Experiment	Density Ratio	Total Frame	Simulation Time (sec/frame)	Rendering Time (sec/frame)
Football	1:800:600	180	51.7	162.1
Container	1:800:500	260	40.3	282.6
Cup	1:800:750	126	34.0	159.1

droplets getting out of the vertical boundaries (the walls and the front transparent glass) are stopped, and then slide down along the surface of the wall. The simulations were performed on a $80 \times 80 \times 80$ grid. Simulation of one time step took about 30-40 seconds. A fixed time step of $\Delta t = \frac{1}{30}$ second was used in all the above 2D and 3D simulations, except for the case of the football. As for the football example, in order to obtain a reasonable visual quality, we adjusted Δt so that the CFL number does not exceed 5.0. Extraction of water surface was done using the marching cube algorithm [Lorensen and Cline 1987], and rendering was done by Mental Ray. The simulation information for each experiment along with the time required to simulate the motion and render the images appears in Table I.

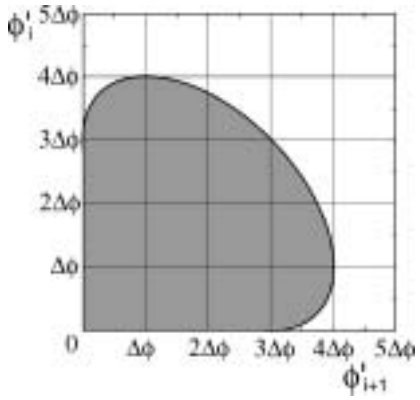

Fig. 8. Plot of (ϕ'_i, ϕ'_{i+1}): the shaded region guarantees monotonic profiles in the case of $\Delta\phi \geq 0$.

8. CONCLUSION

The introduction to the graphics community of the semi-Lagrangian methodology by Stam in 1999 opened the way for the interactive manipulation/control of gaseous fluids. Since then, it has been an open problem to extend the technique to water. In this article, we have proposed a solution to this problem.

The problem mainly consisted of two technical challenges: (1) modeling a multiphase fluid, and (2) finding a dissipation-free technique. In this article, the problem of modeling a multiphase fluid was solved by combining the Navier–Stokes equations with the level set method. The problem of preventing dissipation was solved by two means: (1) adoption of the CIP method, which could model the subcell-level details with third-order accuracy; (2) development of a particle-based method to prevent dissipation in small-scale features such as droplets/bubbles.

Instead of simulating water in a void space, the multiphase fluid formulation proposed here properly simulates the dynamic movement of water, while it engages in complex interactions with surrounding air. Due to the measures taken to prevent dissipation in the proposed method, the simulated water does not unnecessarily lose volume or its motion is not damped to an unphysical extent.

Although "stable but nondissipative water" this article proposes makes a significant improvements in modeling nondissipative behavior of water, the focus has not been put on producing an accurate shape of water surface. When an application requires more accurate modeling of water surface, then Enright et al. [2002] can be adopted to the framework we propose, but with an increased amount of computation. As for modeling droplets/bubbles, to our knowledge, the present work is the first that proposed a mechanism for two-way exchange of masses between the grid-based framework and particles for the conservation of mass and momentum. More accurate modeling of the geometrical shape and size of droplets/bubbles needs further study.

APPENDIX

Monotonic CIP

In this appendix, we develop a method for modifying ϕ'_i and ϕ'_{i+1}, such that the profile becomes monotonic. If, for simplicity, the grid size $(x_{i+1} - x_i)$ is 1, differentiation of Equation (4) produces

$$\Phi'(X) = (3X^2 - 4X + 1)\phi'_i + (3X^2 - 2X)\phi'_{i+1} - (6\Delta\phi)X^2 + (6\Delta\phi)X. \quad (25)$$

When $\Delta\phi \geq 0$, the necessary and sufficient condition for the profile to be monotonically increasing inside the cell is $\Phi'(X) \geq 0$. By manipulating Equation (25), we found that this condition can be reduced to

$$[\phi_i' \geq 0, \quad \phi_{i+1}' \geq 0, \quad \phi_i' + \phi_{i+1}' \leq 3\Delta\phi] \quad OR$$
$$[9\Delta\phi - 6\Delta\phi(\phi_i' + \phi_{i+1}') + (\phi_i' + \phi_{i+1}')^2 - \phi_i'\phi_{i+1}' \leq 0],$$

which corresponds to the shaded region in Figure 8. Similarly, when $\Delta\phi < 0$, the necessary and sufficient condition for the profile to be monotonically decreasing is

$$[\phi_i' \leq 0, \quad \phi_{i+1}' \leq 0, \quad \phi_i' + \phi_{i+1}' \geq -3\Delta\phi] \quad OR$$
$$[9\Delta\phi + 6\Delta\phi(\phi_i' + \phi_{i+1}') + (\phi_i' + \phi_{i+1}')^2 - \phi_i'\phi_{i+1}' \leq 0].$$

Therefore, the monotonic CIP technique works in the following way: when (ϕ_i', ϕ_{i+1}') does not belong to the shaded region, we modify the values so that the tuple goes into the region. Although the modification may create a more diffusive profile than the original one, the result is oscillation-free and still has a third-order accuracy in space.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

Brackbill, J. U., Kothe, D. B., and Zemach, C. 1992. A continuum method for modeling surface tension. *J. Comp. Phys. 100*, 335–354.

Carlson, M., Mucha, R. J., and Turk, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 377–384.

Chen, J. X. and Lobo, N. D. V. 1995. Toward interactive-rate simulation of fluids with moving obstacles using Navier–Stokes equations. *Graph. Models Image Process. 57*, 2, 107–116.

Enright, D., Marschner, S., and Fedkiw, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2002) 21*, 3, 736–744.

Fedkiw, R., Stam, J., and Jensen, H. W. 2001. Visual simulation of smoke. *Comput. Graph. (Proceedings of ACM SIGGRAPH 2001) 35*, 15–22.

Feldman, B. E., O'Brien, J. F., and Arikan, O. 2003. Animating suspended particle explosions. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2003) 22*, 3, 708–715.

Foster, N. and Fedkiw, R. 2001. Practical animation of liquids. *Comput. Graph. (Proceedings of ACM SIGGRAPH 2001) 35*, 23–30.

Foster, N. and Metaxas, D. 1996. Realistic animation of liquids. *Graph. Models Image Process. 58*, 5, 471–483.

Foster, N. and Metaxas, D. 1997a. Controlling fluid animation. In *Comput. Graph. Inter. 97*, 178–188.

Foster, N. and Metaxas, D. 1997b. Modeling the motion of a hot, turbulent gas. *Comput. Graph. (Proceedings of ACM SIGGRAPH '97) 31*, Annual Conference Series, 181–188.

Golub, G. H. and Loan, C. F. V. 1996. *Matrix Computations*. The John Hopkins Univserity Press.

Harlow, F. H. and Welch, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids 8*, 12, 2182–2189.

Kass, M. and Miller, G. 1990. Rapid, stable fluid dynamics for computer graphics. *Comput. Graph. (Proceedings of ACM SIGGRAPH '90) 24*, 4, 49–57.

Lorensen, W. E. and Cline, H. E. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Comput. Graph. (Proceedings of ACM SIGGRAPH '87) 21*, 4, 163–169.

Losasso, F., Gibou, F., and Fedkiw, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 457–462.

McNamara, A., Treuille, A., Popović, Z., and Stam, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 449–456.

Miller, G. and Pearce, A. 1989. Globular dynamics: A connected particle system for animating viscous fluids. *Comput.Graph. 13*, 3, 305–309.

O'Brien, J. and Hodgins, J. 1995. Dynamic simulation of splashing fluids. In *Proceedings of Computer Animation 95*, 198–205.

O'Brien, T. G. G. A. W. B. J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2004) 23*, 3, 463–468.

Osher, S. and Fedkiw, R. 2002. *The Level Set Method and Dynamic Implicit Surfaces.* Springer-Verlag, New York.

Osher, S. and Sethian, J. A. 1988. Fronts propagating with curvature dependent speed: Algorithms based in hamilton-jacobi formulations. *J. Comp. Phys. 79*, 12–49.

Peng, D., Merriman, B., Osher, S., Zhao, H., and Kang, M. 1999. A pde-based fast local level set method. *J. Comp. Phys. 155*, 410–438.

Premože, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. T. 2003. Particle-based simulation of fluids. In *Eurographics 2003 Proceedings*. Blackwell Publishers, 401–410.

Rasmussen, N., Nguyen, D. Q., Geiger, W., and Fedkiw, R. 2003. Smoke simulation for large scale phenomena. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2003) 22*, 3, 703–707.

Sethian, J. A. 1996. Fast marching level set methods for three dimensional photolithography development. *SPIE 2726*, 261–272.

Stam, J. 1999. Stable fluids. *Comput. Graph. (Proceedings of ACM SIGGRAPH '99) 33*, Annual Conference Series, 121–128.

Stam, J. and Fiume, E. 1995. Depicting fire and other gaseous phenomena using diffusion processes. *Comput. Graph. (Proceedings of ACM SIGGRAPH '95) 29*, Annual Conference Series, 129–136.

Staniforth, A. and Côtè, J. 1991. Semi-lagrangian integration scheme for atmospheric model—a review. *Mon. Weather Rev. 119*, 12, 2206–2223.

Sussman, M., Fatemi, E., Smereka, P., and Osher, S. 1998. An improved level set method for incompressible two-phase flows. *Comput. Fluids 27*, 663–680.

Sussman, M., Smereka, P., and Osher, S. 1994. A level set approach for computing solutions to incompressible two-phase flow. *J. Comp. Phys. 114*, 146–159.

Takahashi, T., Fujii, H., Kunimatsu, A., Hiwada, K., Saito, T., Tanaka, K., and Ueki, H. 2003. Realistic animation of fluid with splash and foam. In *Eurographics 2003 Proceedings*. Blackwell Publishers, 391–400.

Terzopoulos, D., Platt, J., and Fleischer, K. 1989. Heating and melting deformable models (from goop to glop). In *Proceedings of Graphics Interface '89*. 219–226.

Treuille, A., McNamara, A., Popović, Z., and Stam, J. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH 2003) 22*, 3, 716–723.

Trottenberg, U., Oosterlee, C., and Schűller, A. 2001. *Multigrid*. Academic Press.

Tsai, Y.-H. R., Cheng, L.-T., Osher, S., and Zhao, H.-K. 2003. Fast sweeping algorithms for a class of hamilton–jacobi equations. *SIAM J. Numer. Anal. 41*, 673–694.

Xiao, F., Yabe, T., and Ito, T. 1996. Constructing oscillation preventing scheme for advection equation by rational function. *Comp. Phys. Comm. 93*, 1–12.

Yabe, T. and Aoki, T. 1991. A universal solver for hyperbolic equations by cubic-polynomial interpolation i. one-dimensional solver. *Comp. Phys. Comm. 66*, 219–232.

Yabe, T., Xiao, F., and Utsumi, T. 2001. The constrained interpolation profile method for multiphase analysis. *J. Comp. Phys. 169*, 556–593.

Received July 2004; revised October 2004; accepted October 2004