Chapter 2
Water and Bubbles

2.1 Animation of Bubbles in Liquid

Abstract This section introduces a new fluid animation technique in which liquid and gas interact with each other, using the example of bubbles rising in water. In contrast to previous studies which only focused on one fluid, this system considers both the liquid and gas simultaneously. In addition to the flowing motion, the interactions between liquid and gas cause buoyancy, surface tension, deformation, and movement of the bubbles. For the natural manipulation of topological changes and the removal of the numerical diffusion, we combine the volume of fluid method and the front-tracking method developed in the field of computational fluid dynamics. Our minimum stress surface tension method enables this complementary combination. The interfaces are constructed using the marching cubes algorithm. Optical effects are rendered using vertex shader techniques.

2.1.1 Introduction

Liquids are very attractive substances. As well as having beautiful optical properties, their movements are mysterious, or as an eastern saying goes, "just observing water can provide good meditation." Many studies have been done in an attempt to animate and render liquids in the computer graphics field. And thanks to recent improvements in computing powers and simulation techniques, more phenomena related to liquids have become subjects of animation.

This section introduces one more subject pertaining to liquid animation, i.e., bubbles.

Bubbles are pockets of air enclosed by liquid and exist everyplace where liquid and air coexist. As opposed to air skimming over liquid surfaces, bubbles are governed by the interactions between air and liquid. There are many factors to be considered when attempting to simulate the deformation and movement of bubbles. There are two flows to consider—i.e., those occurring inside and outside of the bubble bodies. Differences in specific gravity between the two fluids generate buoyancy forces. Surface tension forces are exerted at the interfaces between the two fluids.

C.-H. Kim et al., *Real-Time Visual Effects for Game Programming*,
Gaming Media and Social Effects, DOI 10.1007/978-981-287-487-0_2

In general, the density of liquids is much higher than that of gases. For example, water is eight hundred times heavier than air. This fact is one of the reasons for the free surface approximation in which the existence of air is generally ignored in liquid simulations. Many recent studies on liquid animation have referred to the free surface studies which have been done in computational fluid dynamics (CFD). These studies showed very natural results and some air flows were able to be inserted as surface boundary conditions—e.g., wind. However, since enclosed air is an altogether different affair, those studies are not suitable for bubbles. Besides the additional factors described above, one more consideration needs to be taken into account—i.e., two fluids have to be simulated at the same time. This problem is studied in the form of multiphase flows in CFD with the phase change problem.

Like other fluid problems, many techniques have been developed for the simulation of multiphase flows in CFD. However, since all techniques have their own characteristic approach, in order to decide which technique to use for computer animation, a set of selection criteria are needed. The criteria that we use in this section are the ease of programming, the numerical stability and the fast simulation, even at the cost of accuracy. However, since the main virtue of CFD is accuracy, no existing technique matched these characteristics exactly, therefore we combine and modify various existing techniques for our purposes.

This section presents a new fluid animation technique in which liquid and gas interact with each other, using the example of bubbles rising in water (Fig. 2.1). This system is based on the complementary combination of the volume of fluid (VOF) method and the front-tracking method which were developed in the field of CFD. The VOF method is an efficient and fast scheme for free surface simulation with the inherent capability of topological changes. It can be easily extended for the simulation of multiphase flows. However, to reduce the effect of numerical diffusion in the VOF scheme, the interfaces between the two fluids in the simulation grid need to be decided exactly, which is simple in 2D, but complicated and computationally expensive in 3D with fluid volume constraints. In contrast to the VOF method, the front-tracking method introduces no numerical diffusion. However, a bookkeeping process to maintain the front connectivity is needed to handle the topological changes and physically accurate interfacial geometry is required for the calculation of surface tension. Since our minimum stress surface tension method calculates the surface tension effects not from the interfacial geometry but directly from the simulation data, it was possible to combine these two methods.

Due to the VOF scheme being used, fast interface construction is possible with the marching cubes algorithm. Interfaces composed of polygon meshes are rendered by means of the vertex shader. Optical effects—refract, reflection, and dispersion—are included.

Section 2.1.2 presents the previous works on liquid animation and some related CFD techniques, in order to explain the limitations of previous works and the characteristics of our approaches. Section 2.1.3 introduces some new concepts for the representation of multiphase fluids and overviews our method. In Sect. 2.1.4, the simulation process is discussed in relation to the Navier–Stokes equation. Section 2.1.5 discusses the techniques introduced for visualization. In Sect. 2.1.6, we present our

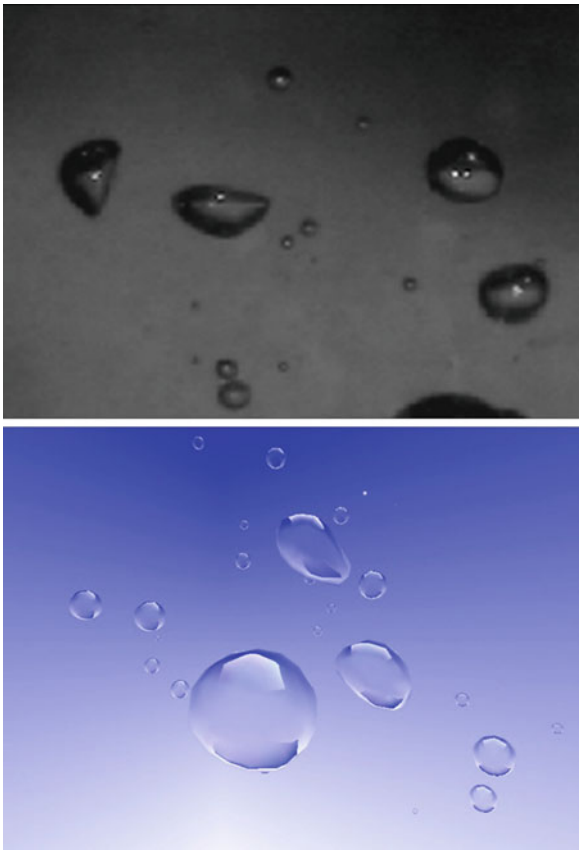

Fig. 2.1 Rising bubbles in liquids. Photo image (*top*) and rendered image (*bottom*)

results. We conclude and discuss ideas for the future research in Sect. 2.1.7. All figures are explained in two dimensions and their extension to three dimensions should be fairly evident.

2.1.2 Previous Work

The characteristics of a physically based model are strongly influenced by the physical and mathematical foundation of that model. Therefore, a combination of both models and CFD techniques is necessary in order to provide a more meaningful explanation. CFD researches include many topics—the accuracy of simulation, numerical techniques, the handling of geometry, and so on. Among them, we will concentrate on only those parts which are directly related to our purpose.

The governing equation of fluids is known as the momentum or Navier–Stokes equations. Following some initial approaches using simplified versions of the Navier–Stokes equations [37, 62], the animation of complex water was studied [19] using the marker and cell (MAC) method [32] with the full 3D Navier–Stokes equation. In the MAC method, the Navier–Stokes equation is discretized within some fixed uniform cells and fluids are expressed by Marker particles. Marker particles were able to describe both natural and detailed scenes [19, 32]. This scheme is also applied to melting animations [5]. In order to treat the smooth and detailed surfaces using these marker particles, the implicit surfaces and level-set methods were used [16]. Realistic optical properties were rendered with the physically based ray tracer. In the simulation of very complex scenes, volume loss occurred and to fix this problem the particle level-set method was introduced [14]. This approach enabled the animation of very complex scenes and velocity extrapolation gave us more control with coarse grids. Although the MAC method presents the explicit expression of liquids with marker particles, it is difficult to estimate the volume of liquid in a cell from marker particles. To represent and simulate two fluids with one grid system, we have to know the volume of each fluid in one grid. Therefore, the animation techniques based on MAC method [5, 14, 16, 19] are not suitable for our purposes.

For fast animation of liquids [43], the volume of fluid (VOF) method [29] was used, in which the liquid surfaces were constructed using the marching cubes algorithm [44] and rendered with polygonal techniques. The VOF method handles topological changes naturally with the marching cubes algorithm, and basically uses only one scalar value—the volume of fluid—for one cell, through which we can know the total volume of fluid in the simulation space. The VOF method assumes that the liquid in a cell is gathered in one corner. From the volume value of one cell and its adjacent cells, the exact position of the liquids needs to be estimated to eliminate the effects of numerical diffusion. This is a problem involving the intersection of a line and a square in 2D cases [67]. In 3D, these become a plane and a cube [23], and some numerical iteration is required in order to find a solution. Therefore, it is inefficient to eliminate numerical diffusion within VOF scheme for computer animation purposes.

Some spherical objects related to fluids such as liquid foams [42], water droplets [17, 89], and soap bubbles [10] have been studied. However for air bubbles enclosed in liquids, the simulation of environmental liquids is unavoidable. This phenomenon can be explained as a kind of multiphase flows. The front-tracking method [80] involves the simulation of multiphase flows without numerical diffusion. The original front-tracking method explicitly discretized the free surface using particles and maintains a connectivity list between these particles [85]. This connectivity list is difficult to maintain when parts of the free surface break apart or merge together as is often seen in complex flows of water and other liquids. To avoid this difficulty, the point-set method was introduced [79]. Although this approach unchains the front-tracking method from its dependence on logical interface point connectivity, the point regeneration algorithm is complex and computationally expensive. The level contour reconstruction method [72] is similar to the combination of the VOF method and the marching cubes algorithm used in [43], which possesses the inherent

capability of being able to deal with topological changes. The feedback from interfaces to simulation grids still removes numerical diffusion. However, for the calculation of surface tension forces and for numerical accuracy, the physically exact interfaces are needed.

Our minimum stress surface tension method is implemented independently from the details of interfacial geometry with the sufficient convergence for computer animation. Moreover, the feedback provided by the front-tracking method removes the numerical diffusion and guarantees mass conservation with the benefits coming from the VOF scheme.

In solving the Navier–Stokes equations, the initial approach was based on explicit finite difference scheme [19, 32]. For computer graphics, the stable fluid scheme [77] based on implicit approaches such as semi-Lagrangian method and implicit diffusion was proposed for large time-steps and numerical stability. Subsequently, efficient pressure iteration was introduced [16]. Our method utilizes these techniques instead of the standard CFD techniques (see Sect. 2.1.4.1).

2.1.3 Overview

2.1.3.1 Representation of Multiphase Fluids

In contrast to previous works which have dealt with the free surface problem, we consider two fluids simultaneously. To represent two fluids with one fixed grid system, we define an *indicator function* $I = I(\mathbf{x}, t)$. $I(\mathbf{x}, t)$ takes the value 1 in one fluid and 0 in the other fluid. A *material field* is defined by the values of I in each cell and Fig. 2.2a shows an example. As you can see in Fig. 2.2a, there are some transition zones between 0 and 1, which are at the interfaces between the two fluids.

We can define the interfaces between two fluids with some isosurface construction algorithm. As is shown in Fig. 2.2b, we used the marching cubes algorithm with a threshold value of 0.5. Details are discussed in Sect. 2.1.5.1.

2.1.3.2 System Outline

In this section, we divide our animation process into three steps in order to provide a conceptual explanation. They are the velocity field update, material field update, and visualization processes. A more detailed explanation of the general processes involved in fluid animation can be found in the literature [16, 19, 29, 32, 77].

Velocity Field Update
In this step, the velocity field is updated from the initial or previous velocity field by solving the Navier–Stokes equation. Material field data are needed for the calculation of the gravity forces and surface tension forces. The details are discussed in Sects. 2.1.4.1 and 2.1.4.2.

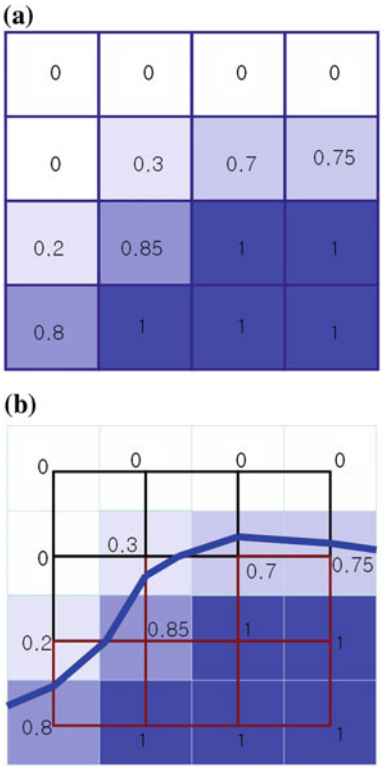

Fig. 2.2 Material field **a** and interfaces **b** constructed from this material field

Material Field Update

After updating the velocity field, we should update the material field by evolving the indicator function to reflect the movement of the fluids caused by the velocity field. This involves the flow of materials. The details are discussed in Sect. 2.1.4.3.

Visualization

From the updated material field, we construct rendering primitives and render them. As well as the polygonal meshes representing the interfaces, some particles are included for the sake of providing more detailed scenes. The details are discussed in Sect. 2.1.5.

2.1.4 Simulation of Multiphase Flows

2.1.4.1 Navier–Stokes Equation

The momentum equation, the so-called Navier–Stokes equation for multiphase flows [80] is

$$\frac{\partial(\rho\mathbf{u})}{\partial t} = \mu\nabla\cdot(\nabla\mathbf{u}) - \nabla\cdot(\rho\mathbf{u}\mathbf{u}) - \nabla P + \rho\mathbf{g} + \int_{\Gamma(t)} \rho\kappa\mathbf{n}\delta(\mathbf{x}-\mathbf{x}_f)ds \tag{2.1}$$

where $\mathbf{u}$ is the velocity, ρ is the density, μ is viscosity, P is the pressure, and $\mathbf{g}$ is the gravity. The surface integral is a surface tension term. The physical definition and finite difference scheme of surface tension are described in Sect. 2.1.4.2.

Conservation of mass written for the entire flow field is

$$\nabla\cdot(\rho\mathbf{u}) = -\frac{\partial\rho}{\partial t}. \tag{2.2}$$

The discrete forms for the finite difference method of Eqs. (2.1) and (2.2) can be written as

$$\frac{\mathbf{w}^{n+1}-\mathbf{w}^n}{\Delta t} = \mathbf{A}^n + \mathbf{F}^{n+1} - \nabla_h P \tag{2.3}$$

$$\nabla_h\cdot\mathbf{w}^{n+1} = M^{n+1}. \tag{2.4}$$

Here $\mathbf{w} = \rho\mathbf{u}$ is the fluid mass flux. The advection, diffusion, and external forces terms in Eq. (2.1) are lumped into $\mathbf{A}$, the right side of Eq. (2.2) is denoted by M, and the surface integral in Eq. (2.1) is denoted by $\mathbf{F}$.

Following the spirit of Chorin's projection method, we split the momentum equation into

$$\frac{\tilde{\mathbf{w}}-\mathbf{w}^n}{\Delta t} = \mathbf{A}^n + \mathbf{F}^{n+1} \tag{2.5}$$

and

$$\frac{\mathbf{w}^{n+1}-\tilde{\mathbf{w}}}{\Delta t} = -\nabla_h P \tag{2.6}$$

where we introduce the variable $\tilde{\mathbf{w}}$, which is the new fluid mass flux if the effect of pressure is ignored. The first step is to find this mass flux using Eq. (2.5)

$$\tilde{\mathbf{w}} = \mathbf{w}^n + \Delta t(\mathbf{A}^n + \mathbf{F}^{n+1}). \tag{2.7}$$

The pressure is found by taking the divergence of Eq. (2.6) and using Eq. (2.4). This leads to a Poisson equation for P

$$\nabla^2 P = \frac{\nabla \cdot \tilde{\mathbf{w}} - M^{n+1}}{\Delta t}, \tag{2.8}$$

which can be solved using a standard Poisson solver. The updated mass flux is found from Eq. (2.6)

$$\mathbf{w}^{n+1} = \tilde{\mathbf{w}} - \Delta t \nabla P. \tag{2.9}$$

The updated velocity is $\mathbf{u}^{n+1} = \mathbf{w}^{n+1}/\rho^{n+1}$.

In this section, the phase change problem coming from heat transfer is not included. In isothermal cases, $\partial\rho/\partial t = 0$, which reduces Eq. (2.2) to

$$\nabla \cdot \mathbf{u} = 0 \tag{2.10}$$

and Eq. (2.5) to

$$\nabla_h \cdot \mathbf{w}^{n+1} = 0 \tag{2.11}$$

with $M = 0$. If we consider Eq. (2.10) as a *volume* conserving condition, the whole process of finding a solution becomes similar to one involving free surface conditions.

Since there is no *vacant* space in our simulation, unlike free surface simulations, all cells should be simulated. Therefore, the free surface conditioning such as classifying cells and modifying the velocities of surface cells [19, 32], is not needed.

In the first projection step of Eqs. (2.5) and (2.7), the stable fluids scheme [77] is incorporated, in which the advection is calculated using the semi-Lagrangian method and the diffusion is calculated with implicit method. The second projection step of Eqs. (2.6), (2.8), and (2.9) is solved in the form of a mass conservation process [16], in which we use a standard conjugate gradient solver as a Poisson solver. All equations are discretized on the standard staggered MAC grids [32].

2.1.4.2 Surface Tension

Surface tension is the apparent interfacial tensile stress (force per unit length of interface) that acts whenever a liquid has a density interface, such as when the liquid is in contacts with a gas, vapor, second liquid, or solid. The mathematical definition of surface tension $\mathbf{F}$ in Eq. (2.1) is

$$\mathbf{F} = \int_{\Gamma(t)} \sigma\kappa\mathbf{n}\delta(\mathbf{x} - \mathbf{x}_f)ds \tag{2.12}$$

where σ is the surface tension coefficient, κ is twice the mean interface curvature, $\mathbf{n}$ is the unit normal to the interface, $\mathbf{x}_f = \mathbf{x}(x, t)$ represents the parameterization of the interface $\Gamma(t)$, and $\delta(\mathbf{x} - \mathbf{x}_f)$ is a three-dimensional delta function that is nonzero only where $\mathbf{x} = \mathbf{x}_f$. Figure 2.3 visually explains the surface tension forces defined in Eq. (2.12). The black lines refer to a portion of the surfaces. Light blue arrows represent the tension forces being exerting at the interfaces. The red arrow,

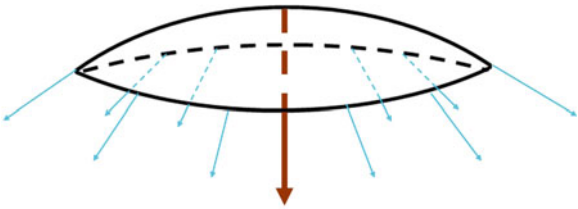

Fig. 2.3 Surface tension forces

representing the sum of these tension forces, represents the total force being exerting on this portion of the surface.

In front-tracking scheme, these forces are calculated using the polygon meshes representing interfaces and distributed to the simulation grids as body forces [72, 80, 85]. Since, the interfaces are constructed from material field discretized on the simulation grids, it is inefficient and unreliable to distribute surface tension forces to the simulation grids estimated from those interfaces. To overcome this inefficiency and remove dependency on interfacial geometry in surface tension calculation as discussed in Sect. 2.1.2, our *minimum stress surface tension method* calculates surface tension forces directly from the material field. The physical meaning of Eq. (2.12) is that the surface tension is a tendency to minimize the total stress of interfacial surfaces. So, we define the stress of material field and let surface tension forces to minimize this stress.

To define the stress of a position on the material field, $S(\mathbf{x})$, first, we define an imaginary stress-zero isosurface whose value is $I^0(\mathbf{x})$, on which $S(\mathbf{x}) = 0$. Then, $S(\mathbf{x})$ can be defined by the deviation of $I(\mathbf{x})$ from $I^0(\mathbf{x})$. In Cartesian coordinate system, $S(\mathbf{x})$ is defined as

$$S(\mathbf{x}) = c \sum_l (I_l^0(\mathbf{x}) - I(\mathbf{x})) \cdot \mathbf{n}_l, \tag{2.13}$$

where c is a control coefficient, l is $\{x, y\}$ in 2D and $\{x, y, z\}$ in 3D, and $\mathbf{n}_l$ is the unit normal of l direction. In our implementation, we define $I_l^0(\mathbf{x})$ as

$$I_l^0(\mathbf{x}) = \sum_l \mathbf{n}_l \cdot \left\{ \sum_{p=m-l} (I(\mathbf{x}_{p+}) + I(\mathbf{x}_{p-})) \right\} / a, \tag{2.14}$$

where $m = \{x, y\}$ and $a = 2$ in 2D, and $m = \{x, y, z\}$ and $a = 4$ in 3D. Finally, we can define the material field version of Eq. (2.12) as

$$F(\mathbf{x}) = -\sum_l (S(\mathbf{x}) \cdot \mathbf{n}_l) \nabla_l I(\mathbf{x}), \tag{2.15}$$

where $\nabla_l I(\mathbf{x}) = (\nabla I(\mathbf{x}) \cdot \mathbf{n}_l)\mathbf{n}_l$.

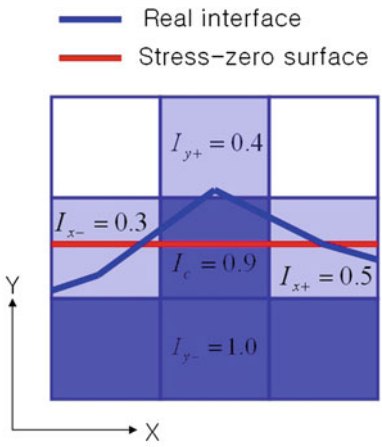

Fig. 2.4 The minimum stress surface tension method

Figure 2.4 shows an example of our method. To find the y portion of $F(center)$, first, we assume an imaginary stress-zero isosurface (red line). In this case, the value of this isosurface, $I_j^0(center)$, is $(I_{x-} + I_{x+})/2 = (0.3 + 0.5)/2 = 0.4$ using Eq. (2.14). The material value of the center cell 0.9 is bigger than 0.4 and this implies that the interfaces constructed by marching cubes algorithm (blue line) would not be on the stress-zero surface. Now, we can calculate the direction and magnitude of the y portion of $F(center)$ using Eq. (2.14). The x portion of $F(center)$ can be calculated in the same way. The extension to 3D cases are fairly evident.

The calculated surface tension forces are inserted to Eq. (2.3) as body forces for Navier–Stokes simulation. Figure 2.5 shows an example. The small lines—the direction is heading from black to white—are normalized surface tension forces inserted as body forces. Red arrows are introduced as visually understandable explanations of the surface tension forces.

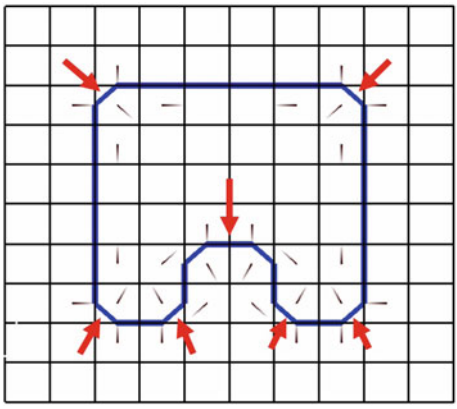

Fig. 2.5 The surface tension forces inserted as body forces

2.1.4.3 Update of Material Field

The last step in the simulation is the update of the material field. As described in Sect. 2.1.3.1, our system describes the positioning of fluids by means of an indicator function. After getting the velocity field as in Sect. 2.1.4.1, we should evolve the indicator function to reflect the movement of the fluids caused by the velocity field.

The time dependence of indicator function I on a velocity field is governed by the equation [29],

$$\frac{\partial I}{\partial t} + u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} = 0. \tag{2.16}$$

In the VOF representation, Eq. (2.16) can be solved by transporting the volume of fluid from one cell to another cell [29]. Through some experiments to get the smoothness of animation in the combination of marching cubes algorithm, we can decide our discretized form of Eq. (2.16). In the case of Fig. 2.6, the change of center cell with our discretization is

$$\frac{\Delta I_C}{\Delta t} = -I_C \cdot v_N - I_C \cdot v_W - I_C \cdot v_E + I_S \cdot v_S. \tag{2.17}$$

While Eq. (2.17) is easy to implement and shows very smooth animation in combination of marching cubes algorithm (will be discussed in Sect. 2.1.5.1), it has the inherent property of numerical diffusion. In ideal simulations, material values of the cells far from interfaces must be 0 or 1. Numerical diffusion occurs when this condition is not fulfilled as shown in Fig. 2.7b. Numerical diffusion prevents the robust and correct liquid simulation. As discussed in Sect. 2.1.2, it is difficult to meet this condition within VOF scheme. However, with the aid of front-feedbacks used in front-tracking method, numerical diffusion can be corrected. In contrast to the MAC representation, we can know the total volume or mass of fluids explicitly with the VOF representation. The total mass of a volume at time t, M^t is

$$M^t = \int_V \rho I^t(\mathbf{x}) dV. \tag{2.18}$$

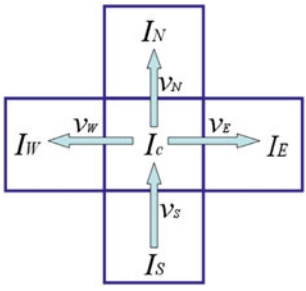

Fig. 2.6 An example of indicator function update

Therefore, what we have to do to correct the mass loss is just to modify $I^{t+\Delta t}$ to meet $M^{t+\Delta t} = M^t$ by changing some material values.

Since we use marching cubes algorithm for constructing interfaces, it is easy to find the location of interfaces or fronts, and move them by scaling adjacent material values. In this modifying step, we fix the value of the indicator function to 0 or 1 *except for* the cells near interfaces or fronts in order to remove numerical diffusion and maintain the location of the interfaces at the same time. Subsequently, we pull out or push back the interfaces to maintain the total mass by scaling the material values near the interfaces. The scaling factor is decided as

$$SF = \frac{M^t - M_{fixed}}{M^{t+\Delta t} - M_{fixed}}. \tag{2.19}$$

Figure 2.7 is a rising bubble example. Figure 2.7a represents initial configurations. Unlike Fig. 2.7b, with our correcting step, Fig. 2.7c shows the numerically perfect mass conservation and no numerical diffusion.

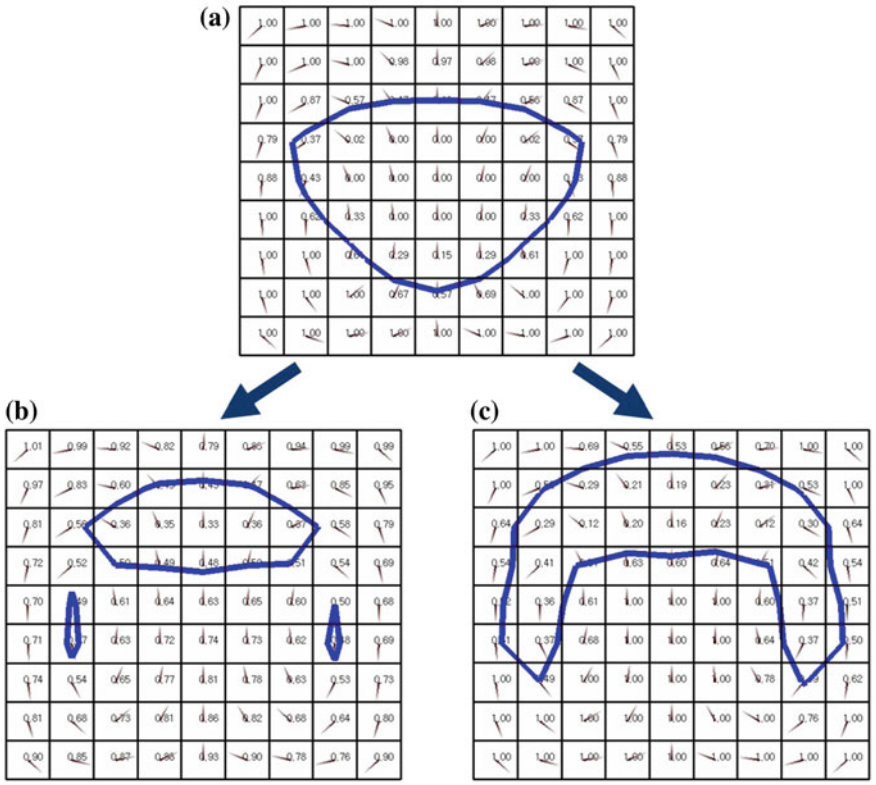

Fig. 2.7 Restricting the numerical diffusion

2.1.5 Visualization

2.1.5.1 Interface Construction

In the front-tracking method, the interfaces between two materials—in our case, water and air—are composed of polygon meshes for the easy calculation of the surface tension. Bookkeeping method [85] in the case of polygon meshes is difficult because of the topological changes of the fluids. Recently, the isosurface construction method for front tracking [72] was used to solve this problem. This approach handles topology changes in natural way, which is appropriate for the purpose of animation. Since our use of surface tension steps using the minimum stress tendencies discussed in Sect. 2.1.4.2 reduces the need for a detailed expression of the interfaces, we were able to use the marching cubes algorithm for isosurface construction. In addition to its compatibility with our staggered grid system, the lookup table style of the marching cubes algorithm supports fast animation [43, 44]. In this case, the material field plays a role of the intensity field needed in marching cubes algorithm (see Fig. 2.2b). The vertex normal was calculated by interpolating the gradient of the material field.

With the marching cubes algorithm, there are many possibilities of discontinuities arising in the animation. Furthermore, since our indicator function is defined by a discontinuous delta function, the continuity of the animation could be damaged. However, though the approach we used to update the indicator function introduces the numerical diffusion without front-tracking steps, it shows smooth animation with the marching cubes algorithm. This is one more benefit which arises from the use of our indicator function update method (discussed in Sect. 2.1.4.3).

2.1.5.2 Particle System

Small bubbles are spherical due to the domination of the surface tension forces. We use the particle system for small bubbles with no deformation. While the computational cost associated with the particles is small, they provide for a lively animation. The velocity of a particle is determined by the linear interpolation of six facial velocities of the cell containing that particle. For natural behavior, buoyant forces are added as body forces, which is similar to the approach taken in the MAC method. Sizes and initial positions are randomly decided. In some cases, the use of the particle system alone could provide for a good animation of bubbles.

2.1.5.3 Rendering

Unlike other approaches using implicit surfaces [14, 16], in our system, the interfaces are composed of polygon meshes, which enables fast rendering supported by hardware acceleration [43]. Some optical effects were able to be implemented by means of a vertex shader. Reflection, refraction, and dispersion effects were applied using conventional vertex shader codes [93], which results in visually pleasing scenes.

2.1.6 Results and Discussion

Following are the results of our animation system implemented using the OpenGL APIs and the NVIDIA vertex shader codes. This system was tested on Windows PC system and the test machine is a PC with 512 MB of RAM and an Intel Pentium IV processor running at 1.4 GHz. It uses an NVIDIA GeForce 2 MX graphics card with 64 MB of video RAM.

Before simulation process, the properties and the initial conditions of fluids should be given. They are the initial velocity field, viscosity and density of each fluid, gravity, surface tension coefficient between two fluids, and initial configuration.

Surface Tension

Figures 2.8 and 2.9 are examples provided to show the convergence of our minimum stress surface tension method. The influence of gravity was omitted for clarity. Even though our algorithm calculates the surface tension forces using material field independently from the details of interface geometry, any arbitrary shapes converged to the spherical ones with no volume loss. In spherical shapes, all surface tension forces are canceled by each other. Some oscillatory phenomena were also included, which are similar to those observed in nature. $13 \times 13 \times 13$ simulation grids were used and frame rate was 7.6 fps.

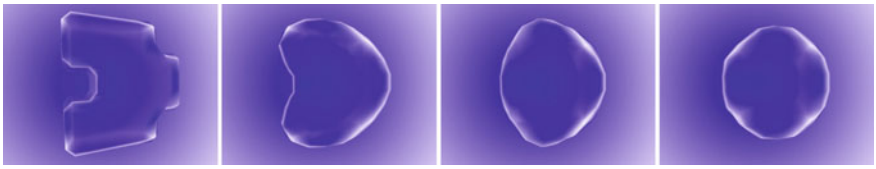

Fig. 2.8 Surface tension convergency

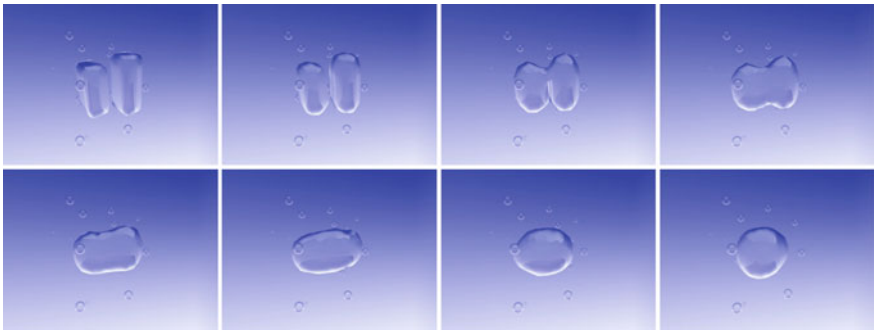

Fig. 2.9 Deformation and merging caused by surface tension

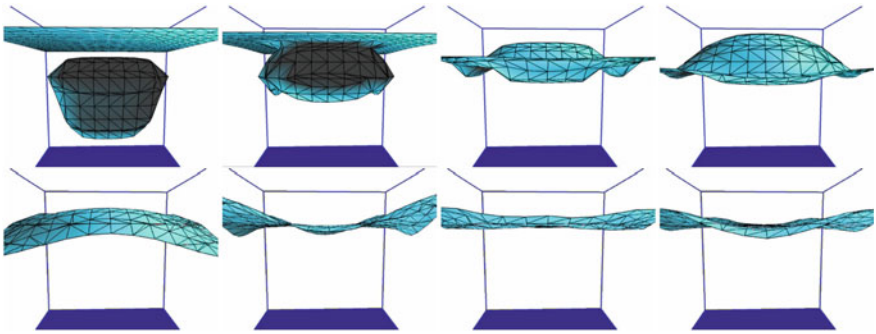

Fig. 2.10 A bubble merges into a free surface

A Bubble Near a Free Surface

When rising bubbles arrive at free surfaces, they are absorbed by the atmospheric air leaving violent impacts on the free surfaces. Figure 2.10 shows this phenomenon. In spite of the severe shape changes, this simulation shows natural animation. The merging of bubble meshes and surface meshes, is done naturally. $15 \times 15 \times 15$ simulation grids were used and frame rate was 5.2 fps.

Although this result proves that it is possible to deal with the free surface condition within our simulation frameworks, there occurs a problem of volume gain of atmospheric air—i.e., the free surfaces lower before meeting the bubble. The reason of this problem is that we conserve *whole air volume or whole liquid volume* in our current implementation as discussed in Sect. 2.1.4.3. To fix this problem, we have to check each separated air volumes—i.e., $M^t = \sum_i M_i^t$—and conserve each of them. With our front-feedbacks, we can easily find the separated volumes and conserve them. This problem with free surface conditions can be handled as a future work.

Rising Bubbles

Figure 2.11 shows a decorated version of the rising bubbles problem. The bubble rises due to their buoyancy. After merging with other small bubble, it rises with certain fixed shape. The shape constitutes a kind of balance point between buoyancy forces and the surface tension forces. The small bubbles are animated using particle system with no deformation as discussed in Sect. 2.1.5.2. Visually pleasing optical effects were included using vertex shader techniques. $9 \times 9 \times 25$ simulation grids were used and frame rate was 1.2 fps.

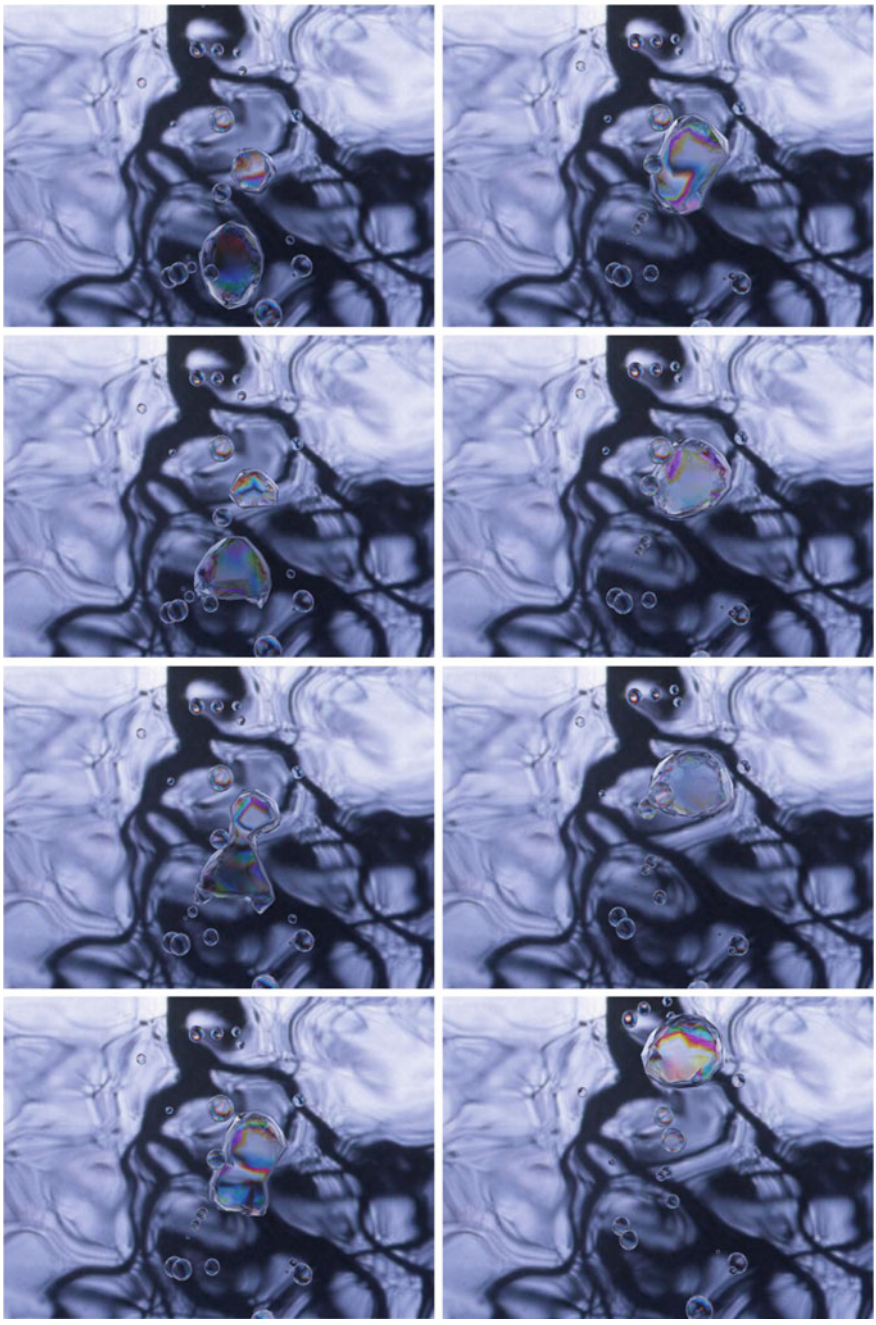

Fig. 2.11 Rising bubbles in a liquid (from *left-up* to *right-down*)

2.1.7 Conclusion and Future Work

In this section, we studied a new fluid animation technique in which liquid and gas interact with each other. This algorithm is based on a complementary combination of various CFD techniques which are selected and modified for computer animation purposes with the aid of our minimum stress surface tension method. The finite difference scheme for the simulation of the multiphase Navier–Stokes equation was introduced and we used appropriate visualization techniques using the marching cubes algorithm and the hardware acceleration.

Since this algorithm can handle topological changes and surface tension fairly easily and with no volume loss or numerical diffusion, we can extend it to the physically based simulation of water droplet model interacting with static environments or other droplets.

2.2 Discontinuous Fluids

Abstract At interfaces between different fluids, properties such as density, viscosity, and molecular cohesion are discontinuous. To animate small-scale details of incompressible viscous multiphase fluids realistically, this section focuses on the discontinuities in the state variables that express these properties. Surface tension of both free and bubble surfaces is modeled using the jump condition in the pressure field; and discontinuities in the velocity gradient field, driven by viscosity differences, are also considered. To obtain derivatives of the pressure and velocity fields with subgrid accuracy, they are extrapolated across interfaces using continuous variables based on physical properties. The numerical methods presented in this section are easy to implement and do not impact the performance of existing solvers. Small-scale fluid motions, such as capillary instability, breakup of liquid sheets, and bubbly water can all be successfully animated.

2.2.1 Introduction

Close-up scenes of splashing water have mysterious attractions. To emphasize the luxurious image of their products in an advertisement, to depict the tense atmosphere before the sword fight in a movie, or just to show off the performance of their brand-new digital camera, many people are trying to catch a moment of this beauty. Recently, the computer graphics community has made great advances in fluid animation, and we are taking one more step toward small-scale realism.

All fluids in our environment are essentially multiphase. This means that property variables are discontinuous at the interfaces between different phases. The small-scale motion of fluids is strongly influenced by these discontinuities. For example,

Fig. 2.12 Capillary instability of a liquid jet; liquid pouring on to a sphere; and bubbly water

the discontinuity of molecular cohesion induces surface tension, which is the phenomenon that smooths out liquid surfaces. It is an interesting fact that surface tension is also responsible for the capillary instability that can break up fluid into small droplets or bubbles. Similarly, discontinuity of density is the reason for Rayleigh–Taylor instability, as well as for buoyancy; and the discontinuity of viscosity influences the shape of air bubbles in water.

In this chapter, we extend previous fluid simulation techniques based on Eulerian grids [14, 16, 47, 77] to incompressible viscous multiphase fluids, focusing on surface tension effects and viscosity changes at both free surfaces and bubble surfaces, as well as on buoyancy. This requires a robust treatment of discontinuities in the pressure and velocity gradient fields. To differentiate them accurately across interfaces, we deploy the ghost fluid method (GFM) of Fedkiw et al. [15], which was developed in computational physics. In combination with the implicit representation of level-set surfaces [63], GFM can treat discontinuities accurately. Surface tension can be modeled using the jump condition in the pressure field, and discontinuities in the velocity gradient field, driven by viscosity differences, can also be considered, permitting subgrid accuracy. Since the numerical methods derived in this section are formulated as simple modifications of previous techniques, they are very easy to implement and do not influence the performance of existing solvers. Results show interesting aspects of small-scale fluid motions such as capillary instability, breakup of liquid sheets, and bubbly water (Fig. 2.12).

2.2.2 Previous Work

The first simulation of the fully three-dimensional Navier–Stokes equation for animating liquids [19] was based on the marker and cell method [32] from computational fluid dynamics. Foster and Metaxas [19] used explicit finite differencing for advection and viscosity, successive over relaxation (SOR) for pressure projection and incompressibility, and massless marker particles for surface representation. Explicit integration methods were subsequently replaced by implicit methods [77] such as semi-Lagrangian advection and implicit viscosity integration, which greatly increased the numerical stability of fluid simulators both for liquid and gas, and

made them easier to implement. Later [16], SOR was replaced by more efficient linear solvers, such as the conjugate gradient method, and the particle-based surface representation was reinforced by implicit level-set surfaces, which greatly improved the smoothness of liquid surfaces and their robustness under topological changes. This hybrid surface representation was enhanced by the particle level-set method [14], which has a much improved mass conservation.

While free surface animation techniques, in which the environmental and enclosed air are ignored, have been extensively developed for liquid animation, the dynamics of multiphase fluids have received less attention. Takahashi et al. [81] reported a multiphase fluid simulator that handles liquid and gas simultaneously, but gave no attention to the dynamic characteristics of liquid–gas interactions. On the other hand, [25] mainly focused on buoyancy and surface tension in their animation of bubbles in liquids. Although their results showed the interesting characteristics of multiphase fluids containing bubbles, it is not clear that their heuristic implementation of surface tension is generally useful. Furthermore, the effect of viscosity differences was not considered, in spite of the large influence of viscosity on bubble shapes. Carlson et al. and Rasmussen et al. [5, 66] used the variational viscosity method to handle thermal changes of viscosity, but they did not consider the large changes that occur across interfaces. More attention to the small-scale features of multiphase fluid was paid by [76]. They demonstrated the characteristics of enclosed air and modeled surface tension using the continuum surface force model [2] that has been generally used in computational fluid dynamics; but Song et al. commented that surface tension effects were not visually significant in their work. We believe that is because they replaced small-scale features by undeformable particles instead of simulating them directly. Similarly, [22] used escaped particles within a particle level-set method to represent air bubbles. A breakthrough was made by [47], who animated the crown phenomenon exhibited by milk by accurately simulating the surface tension of free surfaces. They were able to use a sufficiently large grid, for example 512^3, so that they did not lose small-scale details. However, neither surface tension in bubble surfaces nor viscosity was considered. Their fluid simulator was based on an octree data structure, which is also the basis of the work of this section.

In computational physics, extensive studies have been undertaken to simulate multiphase fluids. For a good survey, [9, 78, 80] and their references are recommended. Although it is difficult to evaluate the techniques reported in those papers from the viewpoint of computer graphics, the work of [36] is found most compatible with the liquid simulation techniques widely used in computer graphics, such as the particle level-set method [14]. The methods used by [36] are motivated by the ghost fluid method [15], and developed using the variable coefficient Poisson equation [45]. In computer graphics, GFM has been used for physically based modeling of fire [61].

Because of the different requirements of computational physics and computer graphics, we separate the pressure jump condition from the density and velocity gradient discontinuities. This is much easier for computer graphics programmers to understand and implement. Although we are approximating the accurate method of [36], our techniques are powerful and robust in animating multiphase fluids with relatively coarse grids.

2.2.3 Overview of Navier–Stokes Simulation

The Navier–Stokes equation for an incompressible viscous fluid is

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla \cdot (\nu \nabla \mathbf{u}) - \frac{\nabla p}{\rho} + \mathbf{f} \tag{2.20}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.21}$$

where $\mathbf{u} = \{u, v, w\}$ is the velocity, ρ is the density, and ν is the (kinematic) viscosity, which is the ratio between the absolute viscosity μ and ρ. The term $\mathbf{f}$ can be used to add external forces such as gravity, buoyancy [20], surface tension forces [25, 76], and control forces [18, 26, 54, 83].

The numerical simulation of Eqs. (2.20) and (2.21) advances by updating the value of $\mathbf{u}$ at the nth time-step, $\mathbf{u}^n$ to $\mathbf{u}^{n+1}$ during a finite time-step Δt. Following Chorin's projection method [4], we discretize Eq. (2.20) by splitting it into two equations with intermediate status $\mathbf{u}^*$:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nabla \cdot (\nu \nabla \mathbf{u}^n) + \mathbf{f} \tag{2.22}$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p}{\rho}. \tag{2.23}$$

To obtain $\mathbf{u}^*$ from $\mathbf{u}^n$, we compute the advection term, $-(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n$, using a semi-Lagrangian method [77], and the viscosity term, $\nabla \cdot (\nu \nabla \mathbf{u}^n)$, using explicit finite differencing or an implicit variable viscosity formulation [5].

The final step is determining $\mathbf{u}^{n+1}$ from $\mathbf{u}^*$. We can write the divergence of Eq. (2.23) as a form of Poisson's equation,[1]

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*, \tag{2.24}$$

since Eq. (2.21) tells us that $\nabla \cdot \mathbf{u}^{n+1}$ should be zero. Once the pressure profile is determined by solving Eq. (2.24), we can get the final velocity profile:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p. \tag{2.25}$$

Buoyancy can be implemented in a simple way by exerting buoyant forces with the spatial constant ρ [20], or more accurately by allowing for the fact that $\rho = \rho(x)$ in the solution of Eq. (2.23) [76]. Since the modeling of buoyancy is not new to the computer graphics community, we will focus on surface tension and viscosity in

[1] If ρ is spatially varying, Eq. (2.24) is only an approximation, but one that is necessary to decouple the pressure jump condition and the density discontinuity.

Sect. 2.2.4. Instead of exerting continuous surface tension forces [76] using $\mathbf{f}$, we use the discontinuity of p in solving Eq. (2.24) to model surface tension effects and thus obtain subgrid accuracy. Viscosity change across the interfaces is also taken into account in solving Eq. (2.22).

One more topic to be considered is the interface tracking method. We use a signed distance function, ϕ, to represent implicitly the interfaces of two immiscible fluids, at least one of which is a liquid. The advection of ϕ, driven by $\mathbf{u}$, can be described by the level-set equation:

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0 \tag{2.26}$$

To solve Eq. (2.26) numerically, we use the semi-Lagrangian particle level-set method [13].

2.2.4 Discontinuous Interfacial Dynamics

We will now describe numerical methods to implement surface tension and viscosity changes at interfaces. Discontinuous variables are extrapolated across interfaces using continuous variables based on their physical properties. This is applied to both the pressure and velocity field, to obtain accurate derivatives at interfaces. Examples are provided in a single dimension for clarity. Extension to two or three dimensions is straightforward.

2.2.4.1 Surface Tension

We will assume the existence of a pressure profile near the interface Γ between two different, immiscible fluids. Using the signed distance function ϕ to represent the geometry of the situation, Γ exists where $\phi = 0$. As shown in Fig. 2.13, surface tension causes a jump J in pressure across Γ, and the magnitude of J is $\sigma\kappa_\Gamma$.

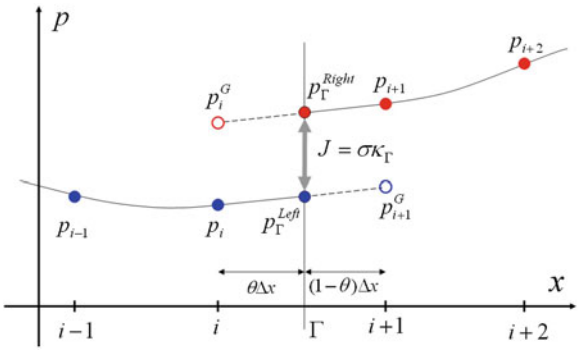

Fig. 2.13 The discontinuous pressure field near an interface Γ

At Γ, σ is the surface tension bcoefficient and κ_Γ is the curvature, which can be determined by interpolating between the curvatures $\kappa = \nabla \cdot (\nabla\phi/|\nabla\phi|)$ of near nodes, using $\theta\kappa_i + (1-\theta)\kappa_{i+1}$, since κ is continuous across Γ due to the implicit surface representation. When κ_Γ is positive, J is positive and vice versa.

The existence of the pressure jump induces a discontinuity in the pressure at Γ, i.e., in Fig. 2.13 the pressure of the left side of the interface, p_Γ^{Left}, and the pressure of the right side, p_Γ^{Right}, are different. This makes it difficult to differentiate p across Γ, in order to discretize Eqs. (2.24) and (2.25) using standard finite differencing. Instead of resolving this problem by smearing out the pressure profile at $i-1, i, i+1, i+2$, we follow the implementation of the variable coefficient Poisson's equation in [45] to keep the profile sharp. First, the pressure at node i, p_i, and the pressure at node $i+1$, p_{i+1}, are extrapolated across Γ to decide the ghost values, p_{i+1}^G and p_i^G :

$$p_i^G = p_i + J \tag{2.27}$$

$$p_{i+1}^G = p_{i+1} - J. \tag{2.28}$$

Using these ghost values, accurate derivatives at Γ can be determined:

$$p_{x,\Gamma}^{Left} = p_{x,i+\frac{1}{2}}^{Left} = \frac{p_{i+1}^G - p_i}{\Delta x} \tag{2.29}$$

$$p_{x,\Gamma}^{Right} = p_{x,i+\frac{1}{2}}^{Right} = \frac{p_{i+1} - p_i^G}{\Delta x}. \tag{2.30}$$

We can discretize Poisson's equation (2.24) at i and $i+1$ as

$$\nabla^2 p_i = p_{xx,i} = \frac{p_{x,\Gamma}^{Left} - p_{x,i-\frac{1}{2}}}{\Delta x} = D(x_i) \tag{2.31}$$

$$\nabla^2 p_{i+1} = p_{xx,i+1} = \frac{p_{x,i+\frac{3}{2}} - p_{x,\Gamma}^{Right}}{\Delta x} = D(x_{i+1}). \tag{2.32}$$

where D represents the right-hand side of Eq. (2.24) in one dimension. Equations (2.31) and (2.32) can be rewritten, using Eqs. (2.29) and (2.30), as

$$\frac{\frac{p_{i+1}^G - p_i}{\Delta x} - \frac{p_i - p_{i-1}}{\Delta x}}{\Delta x} = D(x_i) \tag{2.33}$$

$$\frac{\frac{p_{i+2} - p_{i+1}}{\Delta x} - \frac{p_{i+1} - p_i^G}{\Delta x}}{\Delta x} = D(x_{i+1}) \tag{2.34}$$

and using Eqs. (2.27) and (2.28) as

$$\frac{\frac{(p_{i+1}-J)-p_i}{\Delta x} - \frac{p_i-p_{i-1}}{\Delta x}}{\Delta x} = D(x_i) \tag{2.35}$$

$$\frac{\frac{p_{i+2}-p_{i+1}}{\Delta x} - \frac{p_{i+1}-(p_i+J)}{\Delta x}}{\Delta x} = D(x_{i+1}). \tag{2.36}$$

Fortunately, Eqs. (2.35) and (2.36) can be rewritten as follows:

$$\frac{p_{i+1} + p_{i-1} - 2p_i}{\Delta x^2} = D(x_i) + \frac{J}{\Delta x^2} \tag{2.37}$$

$$\frac{p_{i+2} + p_i - 2p_{i+1}}{\Delta x^2} = D(x_{i+1}) - \frac{J}{\Delta x^2}. \tag{2.38}$$

These equations can be assembled into a linear system $\mathbf{Ax} = \mathbf{b}$, where the matrix $\mathbf{A}$ is symmetric and positive definite with appropriate boundary conditions. Note that the left-hand side terms of Eqs. (2.37) and (2.38) are identical to those used in [16]. A small modification of $\mathbf{b}$ involving the pressure jump, J, is all that is required for accurate implementation of surface tension. This method is applicable to both free surfaces and internal interfaces (bubble surfaces) and can be combined with discretization using an octree data structure [47] without any inconsistency. For free surfaces, the ambient air pressure is used as the Dirichlet boundary condition at nodes which adjoin ambient air, and then the jump condition activates the surface tension effect. Extension to the two- or three-dimensional case is simple. A J term for each coordinate is superposed and then added to or subtracted from $D(\mathbf{x})$, in the same way as in Eqs. (2.37) and (2.38). We can then solve this linear system using the conjugate gradient method with a modified ILU preconditioner [69]. After solving Poisson's equation (2.24), the pressure derivatives of Eq. (2.25) are also determined from Eqs. (2.29) or (2.30), which allows us to compute $\mathbf{u}^{n+1}$.

2.2.4.2 Viscosity

We will assume a velocity profile near an interface Γ between two viscous, immiscible fluids, with viscosities denoted[2] as ν^- and ν^+. Note that the discontinuity of viscosity brings about a discontinuity of velocity gradient field across Γ, even though velocity is continuous at Γ. In general, the exact velocity at the interface, u_Γ, cannot be stored in a Eulerian grid. This means that we cannot determine the exact derivatives of u across Γ, which leads to errors in the viscosity step of Eq. (2.22), which will be propagated to adjacent grid elements. Larger differences in viscosity will cause more serious errors.

To resolve this problem, the ghost values, u^G_{i+1} and u^G_i, are obtained by extrapolation u_i and u_{i+1} across u_Γ, as shown in Fig. 2.14. Then we can express the exact

[2]The notation + or − originated from the level-set representation of the interfaces. We use $\phi < 0$ regions for water and $\phi \geq 0$ regions for air. Instead of + or −, "left" or "right" was used in Sect. 2.2.4.1 since the sign of the pressure jump is not dependent on the sign of ϕ.

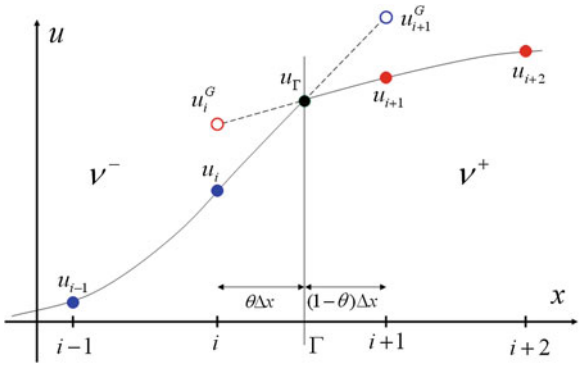

Fig. 2.14 Velocity profile near an interface

derivatives just left and right of Γ as

$$u^-_{x,\Gamma} = u^-_{x,i+\frac{1}{2}} = \frac{u^G_{i+1} - u_i}{\Delta x} \tag{2.39}$$

$$u^+_{x,\Gamma} = u^+_{x,i+\frac{1}{2}} = \frac{u_{i+1} - u^G_i}{\Delta x}, \tag{2.40}$$

while the standard finite difference formulation is

$$u_{x,i+\frac{1}{2}} = \frac{u_{i+1} - u_i}{\Delta x}. \tag{2.41}$$

Now we can use the known physical properties of interfaces. First, the velocity should be continuous:

$$u^-_{x,\Gamma}\theta\Delta x + u^+_{x,\Gamma}(1-\theta)\Delta x = u_{x,\Gamma}\Delta x. \tag{2.42}$$

And, due to the no-slip condition, the viscous acceleration should be the same on both sides of the interface:

$$\nu^- u^-_{x,\Gamma} = \nu^+ u^+_{x,\Gamma}. \tag{2.43}$$

After rearranging Eqs. (2.42) and (2.43), we can rewrite $\nu^- u^-_{x,\Gamma}$ and $\nu^+ u^+_{x,\Gamma}$ as

$$\nu^- u^-_{x,\Gamma} = \hat{\nu} u_{x,i+\frac{1}{2}} \tag{2.44}$$

$$\nu^+ u^+_{x,\Gamma} = \hat{\nu} u_{x,i+\frac{1}{2}}, \tag{2.45}$$

where the effective viscosity[3] $\hat{\nu}$ is $(\frac{\theta}{\nu^-} + \frac{1-\theta}{\nu^+})^{-1}$. We note that it is unnecessary to decide the values of u_i^G and u_{i+1}^G, since the discontinuity is not in the value of u but in that of u_x.

The diffusion equation,

$$u^{new} = u + \nabla \cdot (\nu \nabla u) \Delta t, \tag{2.46}$$

is part of Eq. (2.22); we can discretize it at node i as

$$u_i^{new} = u_i + \frac{\nu^- u^-_{x,\Gamma} - \nu^- u_{x,i-\frac{1}{2}}}{\Delta x} \Delta t. \tag{2.47}$$

Using Eq. (2.44), we can rewrite this equation as

$$u_i^{new} = u_i + \frac{\hat{\nu} u_{x,i+\frac{1}{2}} - \nu^- u_{x,i-\frac{1}{2}}}{\Delta x} \Delta t. \tag{2.48}$$

After repeating a similar process at node $i+1$, we obtain two equations which express the velocities near Γ:

$$u_i^{new} = u_i + \frac{\hat{\nu} \frac{u_{i+1} - u_i}{\Delta x} - \nu^- \frac{u_i - u_{i-1}}{\Delta x}}{\Delta x} \Delta t \tag{2.49}$$

$$u_{i+1}^{new} = u_{i+1} + \frac{\nu^+ \frac{u_{i+2} - u_{i+1}}{\Delta x} - \hat{\nu} \frac{u_{i+1} - u_i}{\Delta x}}{\Delta x} \Delta t. \tag{2.50}$$

High viscosities, large differences, and large time-steps are all troublesome in this explicit scheme. Therefore, we rewrite Eqs. (2.49) and (2.50) in implicit form:

$$-\lambda \nu^- u_{i-1}^{new} + (1 + \lambda \hat{\nu} + \lambda \nu^-) u_i^{new} - \lambda \hat{\nu} u_{i+1}^{new} = u_i \tag{2.51}$$

$$-\lambda \hat{\nu} u_i^{new} + (1 + \lambda \nu^+ + \lambda \hat{\nu}) u_{i+1}^{new} - \lambda \nu^+ u_{i+2}^{new} = u_{i+1}, \tag{2.52}$$

where $\lambda = \Delta t / \Delta x^2$.

These equations form two rows of a linear system $\mathbf{Ax} = \mathbf{b}$. This is a modification of the single-phase case of [77], but nevertheless $\mathbf{A}$ is still symmetric and positive definite, in the same manner as the variable viscosity of [5]. When implemented, based on an octree data structure, the unpreconditioned conjugate gradient method showed better performance than preconditioned methods when the initial guess for $\mathbf{x}$ was $\mathbf{u}^n$. Using an adaptive grid, the terms Δx in Eq. (2.39) and in Eq. (2.47) are different. The errors caused by T-junctions [47] do not significantly impact the visual results.

[3] This approach separates the pressure jump condition from the viscosity gradient jump condition. For a more complicated and coupled treatment, see [36].

2.2.5 Results

Let us explain the small-scale features of fluid motion simulated by the techniques just described with examples. The first example is the capillary instability of the liquid jet in Fig. 2.15. A 5 mm diameter jet is introduced into the computational domain from the left and gravity accelerates it rightward. First, the head of the jet becomes rounded by surface tension. As the head gets bigger, necking develops, and then the head is pinched off. This is the start of instability. This process is propagated to the liquid following, causing repeated sequential pinching-off. The oscillation of drops due to surface tension makes them look like elastic balls. The very thin and short filaments left after breaking also coalesce into small droplets, but some of them are lost, as reported by [21]. Refer to Picture 122 and 123 in [11] as a comparison of the simulated results with a real picture.

Similar phenomena are found in a liquid sheet, as shown in Fig. 2.16. Surface tension rounds the edges of a thin liquid sheet, then forms it into pipes, which tear the sheet. The torn parts coalesce with adjacent regions of the original sheet, but some fragments are lost because they are so thin. These phenomena also appear in photographs, such as Picture 149 in [11]. Some more interesting scenes are shown in Fig. 2.17. Here, liquid is poured on to a static sphere which causes it to spread into sheets. These are agglomerated by surface tension, finally forming many drops. The strong surface tension makes the liquid sheet behave as an elastic membrane.

Fig. 2.15 The capillary instability of a liquid jet. The effective resolution is 64^3 by 25

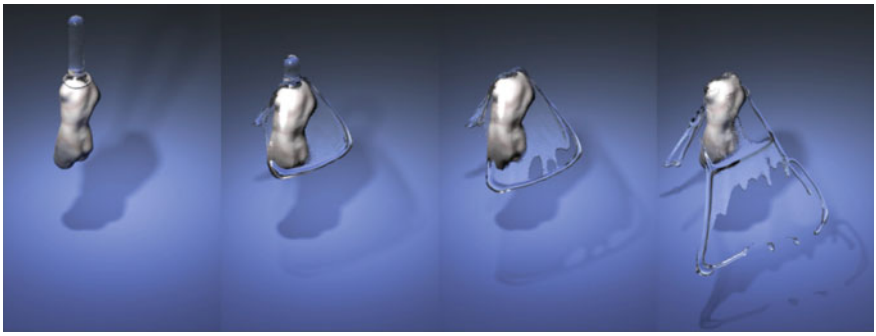

Fig. 2.16 The breakup of a liquid sheet. The effective resolution is 512^3

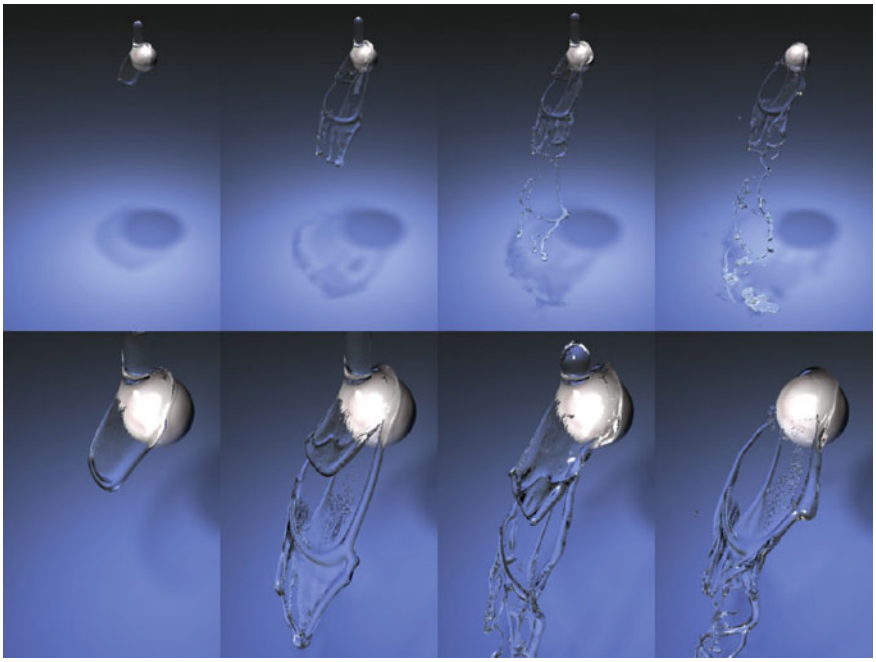

Fig. 2.17 A small-scale scene of liquid pouring. The effective resolution was 512^3 by 2

Figure 2.18 shows an animation of bubbly water. Air is introduced from the bottom and a static sphere disturbs its flow. The breakup of air, the formation and rising of bubbles, and explosions at the surface are all animated. The phenomena simulated in previous examples are also seen in this animation, but viscosity has now become significant. It is difficult to get visually pleasing bubbles without considering the viscosity jump, because the ratio between the viscosities of two fluids influences the position of the buoyant vortex center which affects bubble shapes as much as surface tension.

These simulations were performed on a desktop PC with 3.4 GHz CPU and 2 GB RAM. For the octree data structure, the implementation of [46, 47] was followed. Each time-step took at most 2 min of computation time with an average of 1 min. At most, 20 time-steps per frame were used, allowing one example sequence to be generated in two days of computing. This method is not primarily intended to compete with existing techniques in terms of efficiency; however, it is known that surface tension effects can induce surface oscillations when large time-steps are used, thus slowing the simulation as a whole. However, what limits the time-step in our experience is not the surface tension, but the swirling near very small droplets or bubbles.

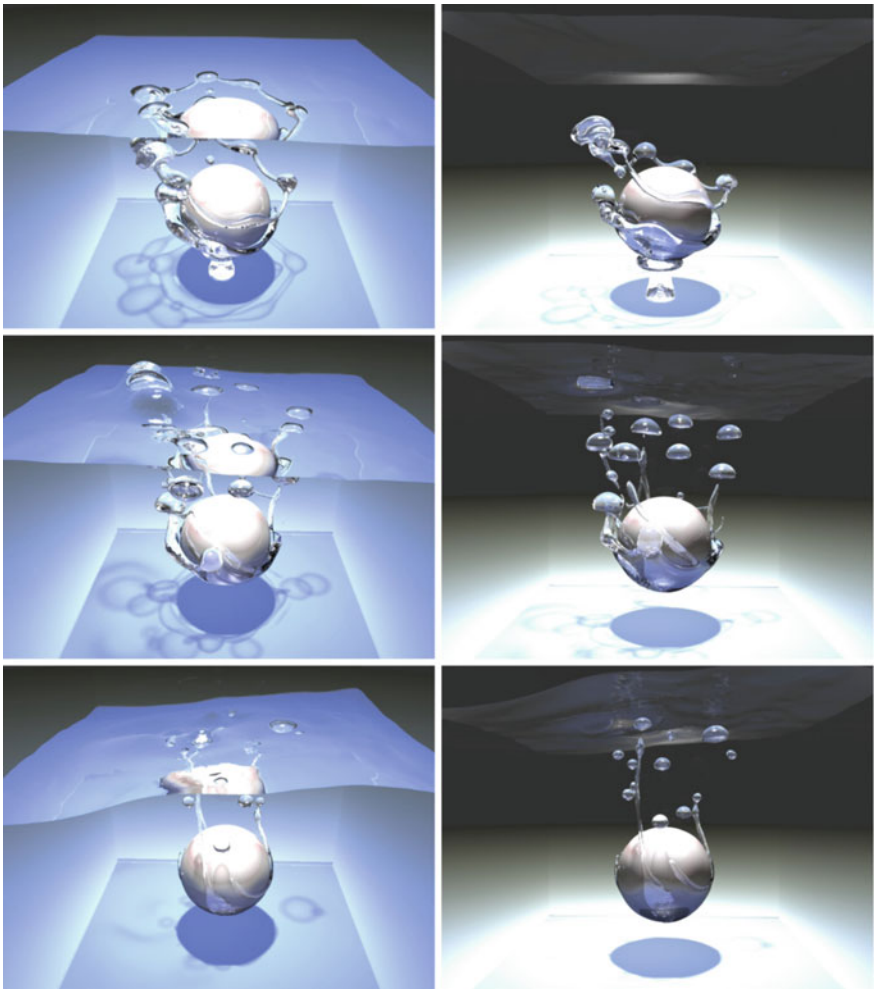

Fig. 2.18 The animation of bubbly water. The effective resolution was 256^3

2.2.6 Conclusion

This section has described a technique for animating fluids which have a discontinuity in their state variables. We have extended previous techniques to multiphase fluids with surface tension effects and viscosity changes at their interfaces, as well as modeling buoyancy. Discontinuities in the pressure and velocity gradient fields were treated in a sharp fashion which preserved subgrid details. The resulting numerical methods are easy to implement and do not influence the performance of existing solvers. Based on these techniques, we have been able to show new aspects of small-scale fluid motions.

One technical extension of this work would be to consider the discontinuity of velocities tangent to interfaces. And also this method of modeling viscosity could be enhanced to include elastic or plastic bodies. An efficient shape control algorithm for multiphase fluids would be another interesting project.

2.3 Bubbles Alive

Abstract This section proposes a hybrid method for simulating multiphase fluids such as bubbly water. The appearance of subgrid visual details is improved by incorporating a new bubble model based on smoothed particle hydrodynamics (SPH) into a Eulerian grid-based simulation that handles background flows of large bodies of water and air. To overcome the difficulty in simulating small bubbles in the context of the multiphase flows on a coarse grid, we heuristically model the interphase properties of water and air by means of the interactions between bubble particles. As a result, we can animate lively motion of bubbly water with small-scale details efficiently.

2.3.1 Introduction

The lively but chaotic motion of bubbles has enchanted and challenged many scientists. Besides the engineering applications, including ship hydrodynamics, cooling of nuclear reactors, and laundry machines, an understanding of bubbles is indispensable to the visual realism of computer-generated animations that show the multiphase characteristics of fluids. In computer graphics, many researchers are struggling to get more realistic bubbles and foams by means of physics-based fluid animation, powered by computational fluid dynamics.

The two major approaches, based on Eulerian grids and Lagrangian particles, have been competing with each other, but are now being combined. This is desirable because they are complementary methods: a particle system based on smoothed particle hydrodynamics (SPH) can be much more flexible and controllable if it concentrates on small-scale details, while large bodies of water and air can be handled efficiently and faithfully by a grid-based solver, without requiring excessive resolution.

The hybrid approach to multiphase flows (including bubbles) has received less attention than the simulation of splashes and droplets, because the difference in scale as compared to the background flow is more severe for bubbles than droplets. Water dominates the inertia because its density is 800 times higher than that of air, and thus bubbles require the surrounding water to be simulated in more detail than the air around a splash. In our experience, each bubble should occupy at least 3^3 nodes (or 3^2 in 2D) to have numerical meaning. This makes it infeasible to refine the grid sufficiently to capture all the small details, especially in a graphics context. That is

why it is desirable to develop a dynamic model appropriate for representing details at the subgrid scale.

This section proposes a hybrid method for simulating multiphase fluids, especially focusing on bubbles. To avoid excessive refinement of the background grid, while maintaining the subgrid details of bubble motion including path instability, we model the interphase properties of water and air in terms of the interactions between bubble particles. While this is ultimately a heuristic approach, it is underpinned by the SPH vorticity confinement method and an analysis of the cohesive forces that generate subgrid turbulence. This combination enables us to capture the natural look of moving bubbles in a way that harmonizes with an underlying grid-based simulation of multiphase flows.

2.3.2 Previous Work

The success of grid-based liquid animation techniques that use a free surface single-phase model (see [14, 16, 19] for examples) led to work on the direct numerical simulation of multiphase phenomena [25, 27, 31, 39, 41, 51, 57, 76].

Premoze et al. [65] presented a particle-based method for fluid simulations that can handle multiphase liquids. Müller et al. [60] applied the SPH method to multiple phases, and [6] modeled the nucleation, collision, and drag interactions of bubbles and foams, based on a background SPH simulation.

Kim et al. [35] used the SPH method to model escaped particles within the particle level-set method [12], so as to resolve subgrid splashes. Losasso et al. [52] improved this approach by coupling a model of dense water volume to diffuse sprays. Greenwood and House [22] also modeled escaped particles to give a more detailed look to bubbles and foams, but without using SPH. Thüerey et al. [84] coupled SPH bubbles to shallow water simulations using locally defined vortices on particles.

2.3.3 A Hybrid Approach

We use the Eulerian method to model the background motions of water and air bodies which are large enough to be captured using a simulation grid which can be managed by an ordinary single-CPU computer. The bubbling details that are too small to be handled on such a grid are simulated by SPH particles. We build our system on the particle level-set fluid solver [14] in order to generate bubble particles by incorporating the escaped particles back into the SPH system as bubbles, similar to [22]. However, this hybrid framework and our bubble model would also integrate well with other grid-based techniques such as the CIP method [76], the BFECC method [39], the CLSVOF method [57], or the Lattice Boltzmann method [82], if appropriate ways of generating bubbles were available.

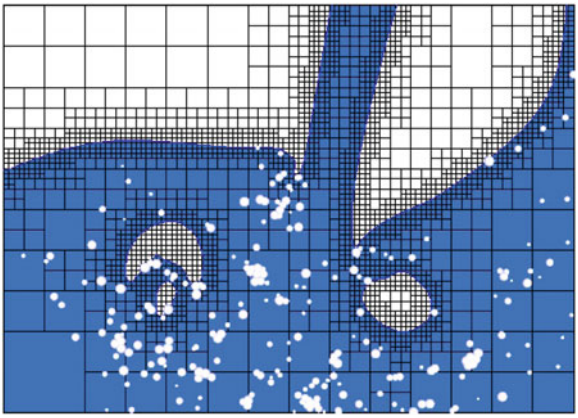

Fig. 2.19 A schematic outline of our hybrid system. The body of water is colored *blue* and *bubble* particles are drawn as *white circles*

Figure 2.19 is a schematic overview of our hybrid system. Since we accelerate our solver by using an octree grid [47], the scale difference between the grid spacing and the particle radius is large. This difficulty is resolved by our subgrid-scale bubble dynamics, which we develop in Sect. 2.3.4.

2.3.3.1 Grid-based Background Simulation

The Navier–Stokes equations describing inviscid incompressible fluid motion are

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p/\rho = \mathbf{f} \tag{2.53}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.54}$$

where $\mathbf{u}$ is the velocity, p is the pressure, ρ is the density, and $\mathbf{f}$ is the aggregate of the external forces including gravity and the momentum exchange from the SPH bubbles that occur during coupling. Since numerical methods of solving Eqs. (2.53) and (2.54) are well known, we refer readers to [14, 27, 47] for details.

2.3.3.2 SPH Overview

The acceleration of a particle i is determined by a sum of forces exerted by adjacent particles, $\mathbf{f}_{ij}$, as follows:

$$\mathbf{a}_i = \sum_j \mathbf{f}_{ij}/\rho_i, \tag{2.55}$$

where the density of a particle i is defined as $\rho_i = \sum m_i W(\mathbf{x}_{ij}, r_i)$. We use the radially symmetric kernel functions $W(\mathbf{x}, r)$ with support r, as defined in [59]. The velocity

and the position of a particle can be determined by sequential Euler integrations such as $\mathbf{v}^{t+\Delta t} = \mathbf{v}^t + \mathbf{a}^t \Delta t$ and $\mathbf{p}^{t+\Delta t} = \mathbf{p}^t + \mathbf{v}^{t+\Delta t} \Delta t$, where Δt is a time-step. Following the adaptive radius approach of [1], which provides a versatile description of bubble details, the pressure force can be expressed as

$$\mathbf{f}_{ij}^{pressure} = -V_i V_j (P_i + P_j)(\nabla W(\mathbf{x}_{ij}, r_i) + \nabla W(\mathbf{x}_{ij}, r_j))/2, \tag{2.56}$$

where the volume V_i is m_i/ρ_i, r is the radius, the mass m_i is proportional to r_i^3, $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$, and the pressure $P_i = k\rho_i$ with a control parameter k. In general, SPH systems largely depend on viscosity, especially to improve stability when they are used to simulate large bodies. Since we use a grid-based solver to deal with large bodies, the viscous forces can be omitted.

2.3.3.3 Two-way Coupling

The major coupling forces which make the bubble particles follow the background flows are drag and lift forces [6, 53], given by

$$\mathbf{f}_i^{drag} = -k_{drag} r_i^2 |\mathbf{v}_i - \mathbf{u}_i| (\mathbf{v}_i - \mathbf{u}_i) \tag{2.57}$$

$$\mathbf{f}_i^{lift} = -k_{lift} V_i (\mathbf{v}_i - \mathbf{u}_i) \times \omega_i, \tag{2.58}$$

where $\mathbf{u}_i$ and $\omega_i = \nabla \times \mathbf{u}_i$ are the velocity and the vorticity, which are interpolated at $\mathbf{p}_i$ from the grid values. Initially, we tried to simulate a simulation for the path instability of bubbles with lift forces, but this did not work well enough since Eq. (2.58) relies on the vorticity field around $\mathbf{p}_i$ being highly refined. This is one of the motivations to develop the heuristic bubble model of Sect. 2.3.4.

The forces reacting to these coupling forces are transferred to the surrounding fluid through Eq. (2.53) after being distributed across a number of adjacent nodes. We also use reaction forces to model the popping of bubbles when they merge with the ambient air. In many cases, the SPH time-step needs to be smaller than the grid simulation time-step. Since the reaction forces change the grid velocities and repeated updating makes their values diverge, they must be stored separately and only added to the right-hand side of Eq. (2.53) once per grid simulation time-step.

2.3.4 Bubbles

2.3.4.1 SPH Vorticity Confinement

Unlike droplets moving through ambient air, bubble particles are subject to strong velocity diffusion because they are coupled to the surrounding fluid by drag and lift forces. Furthermore, these forces are determined from values interpolated on the

coarse grid. To simulate the motion of bubbles in more detail, we therefore introduce a heuristic representation of the vorticity confinement [20] into the SPH method.

First, we measure the vorticity $\omega = \nabla \times \mathbf{v}$ at the mass center of two SPH particles, $\mathbf{p}_\oplus = (m_i \mathbf{p}_i + m_j \mathbf{p}_j)/(m_i + m_j)$. In contrast to the grid-based method of [20], we are able to express the vorticity location vector η as $\eta = \mathbf{p}_\oplus - \mathbf{p}_i$. We can use a normalized version of η, $\mathbf{N} = \frac{\eta}{|\eta|}$, to determine the confinement force:

$$\mathbf{f}_{ij}^{vorticity} = \varepsilon \left(\mathbf{N} \times \frac{\omega}{|\omega|} \right) \rho_i . \tag{2.59}$$

The original vorticity confinement method used by [20] can amplify the existing vorticity over time because the incompressibility enforced by the projection step ensures stability. Taking a similar approach makes the SPH system diverge and we therefore use a normalized ω.

2.3.4.2 Cohesive Forces

Due to the very large density ratio of water to air, water exerts a high pressure on air bubbles causing them to merge rapidly. To achieve a physically accurate simulation, multiphase SPH methods such as those of [24, 60] are desirable. However, because we simulate the water on a coarse grid, we have to take care of the multiphase interactions without explicit models of water particles or detailed velocities around air particles. By assuming that air particles are surrounded by water except where they are explicitly modeled, we can handle this multiphase property by simulating the attraction forces between touching particles, rather than attempting to model the forces exerted by water particles on air particles. Finally, we introduce a cohesive attraction force between particles:

$$\mathbf{f}_{ij}^{attraction} = k_{attraction} W_{attraction}(\mathbf{x}_{ij}, r_i + r_j)\rho_i . \tag{2.60}$$

We use a constant-valued function for $W_{attraction}$ to make it easy to establish a force that balances the pressure forces. The pressure force in Eq. (2.56) pushes adjacent particles outward when the density ρ_i becomes high due to the attraction forces. Becker and Teschner [3] introduced a similar force to represent surface tension, but this can be adequately modeled by the intrinsic properties of SPH in the physical situation with which we are dealing.

One way of inducing clustering would be to use the pressure kernels of [1] or [3] with a negative term so that attraction forces are exerted on adjacent particles when the particle density is low. This is a reasonable approach, but our attraction force will be physically more plausible for bubbles under large water pressure. It also works better with the vorticity confinement techniques explained in the previous section, since it can suppress the scattering of particles by centrifugal effects to the extent required.

2.3.4.3 Subgrid Turbulence

The beauty of bubble motions is mainly a result of their unstable paths. Even a single bubble rising in calm water moves along a zigzag or spiral path due to its own wake (see [74] for an example). Our combination of a cohesive attraction force and SPH vorticity confinement approximates this characteristic motion (see Fig. 2.20) when two or more particles are close together. For single bubbles, it is simplest to add disturbances to the particles' velocities based on random numbers. This also helps to generate an initial vorticity and our system generates the natural look of turbulent bubble motion with the combination of these techniques.

2.3.4.4 Buoyancy

The rising velocities of bubbles are determined by the balance between drag and buoyancy, which establishes a terminal upward velocity. We generally make the buoyant force $\mathbf{f}_{buoyancy}$ proportional to the volume of each particle. An alternative is to make the buoyant force proportional to the difference between the current velocity and a terminal velocity approximately proportional to the particle radius [55]. This could be used to improve the upward motion of the bubbles.

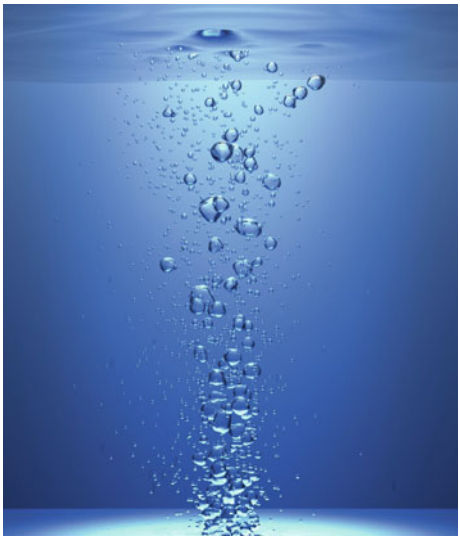

Fig. 2.20 Rising bubbles in calm water. This example shows the realistic motion of bubbles generated by our bubble model coupled to background flows. Simulation took 3 h on an octree grid with an effective resolution of 256×128^2. A maximum of 2,600 SPH particles were used

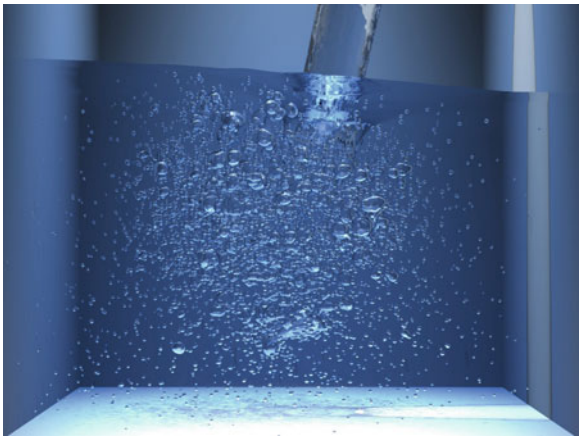

Fig. 2.21 Water pouring with turbulent multiphase bubble flows. Simulation took 6 h on an octree grid with an effective resolution of 256×128^2. A maximum of 8,000 SPH particles were used

2.3.5 Examples

Both water surfaces and bubbles can be ray traced as a single level-set surface by performing on-the-fly Boolean operations that subtract air bubbles from water bodies. The particle radii were set between 0.3 and 0.8 of the grid spacing. Simulations were performed on a PC with an Intel Core2 CPU running at 3 GHz.

Figure 2.20 shows bubbles freely rising in water. In this example, bubble particles are seeded randomly at the bottom and then rise, demonstrating the basic capabilities of our bubble model. The lively and natural motion of bubbles, including flickering, merging, separation, and spiral path instability were simulated successfully. On the accompanying video there are animations with and without our vorticity confinement heuristic, which show that simply adding random disturbances is not adequate. Our bubbles pop as soon as they reach the surface, rather than persisting as foam, which could be implemented using the methods already investigated by [6, 22]. Figure 2.21 shows water being poured.

The atomization of large bodies of air is naturally modeled by escaped particles, and the coupled motion of level-set surfaces and SPH particles achieves realistic bubbly water.

2.3.6 Conclusion

This section has presented a hybrid of Eulerian grid-based simulation and Lagrangian SPH for the realistic simulation of multiphase fluids, focusing on bubbles. Using this heuristic bubble model, we can generate natural looking computer-generated bubbly water.

2.4 Hybrid Simulation of Miscible Mixing with Viscous Fingering

Abstract In this section, we simulate solids and liquids dissolving or changing to other substance by modeling mass transfer phenomena, and deal with the very small-scale phenomena that occur when a fluid spreads out at the interface of another fluid. We model the pressure at the interfaces between fluids with Darcy's Law and represent the viscous fingering phenomenon in which a fluid interface spreads out with a fractal-like shape. We use hybrid grid-based simulation and smoothed particle hydrodynamics (SPH) to simulate intermolecular diffusion and attraction using particles at a computable scale. As a result, we animate fluids mixing and objects dissolving.

2.4.1 Introduction

In computer graphics, many fluid simulation techniques have been developed and used to create realistic animations. However, most of those techniques focus on immiscible fluids such as water, air, and bubbles. Losasso et al. [51] simulated fire and more than two liquids in the same scene, but did not deal with miscible fluids. Recently, Zhu et al. [91], Mullen et al. [58], and Park et al. [64] have presented miscible fluid simulations. However, they excluded the physical and chemical phenomena in which fluids are mixed and react with each other. When two different fluids meet, they spread out in a fractal shape because of physical pressure differences and diffusion laws. We can see this happen when ink is dropped into water. Substances can also melt and be dissolved by mass transfer caused by chemical reaction, and then change into other substances. Molecules of solute float about in the flowing fluid and spread out in a complicated fashion. This section proposes methods of simulating complicated fluid phenomena like those described above, and present animations of the interaction of miscible fluids such as ink, water, bubbles, and melting solids.

Stam [77] demonstrated stable fluid simulation using a semi-Lagrangian advection method and a decomposed version of the Navier–Stokes equation. Following Foster and Fedkiw [16], many researchers have developed multiphase fluid simulation techniques that use a level-set method. These techniques are used by the special effects industry and help to produce movies, advertisements, and games. The technique, in this section, is built on these existing technologies, with the aim of achieving stability and straightforward implementation. Other researches on miscible fluids [58, 64, 91] have used the lattice Boltzmann method (LBM), the density-based weighted essentially non-oscillatory (WENO) method, or the phase field method (PFM), but we employ none of these.

We track the interfaces of a large number of fluids using a similar approach to multiple level-set techniques. We use a separate level-set for each separate substance, and name the interface where the substances mix the mixing surface. At the mixing

surfaces defined by these multiple level-sets, intricate mixing phenomena occur that create complicated fractal shapes because of the differences in concentration, viscosity, and pressure between the different substances. We assume that these phenomena are sufficiently like the flow of liquids through porous media, which follows Darcy's Law, and we model the viscous fingering exhibited by mixing fluids and simulate it using the ghost fluid method (GFM). The viscous fingering model proposed in this section is simple and easy to implement since it is modeled with pressure jump. We also model the mass transfer phenomena caused by chemical reactions using the equation of heat-dependent mass transfer proposed by Mihalef et al. [57] and Son et al. [71]. These techniques allow us to animate substances that change phase and melt to form other substances.

Hong et al. [28] and Losasso et al. [52] simulated detailed splashing and bubble motion by combining a grid-based version of Euler's method with a particle-based Lagrangian approach. We simulate the motion of molecule-like particles that represent a concentration using this hybrid method. These concentration particles experience forces that include diffusion and quasi-intermolecular attraction and repulsion. We control and simulate these forces simply using smoothed particle hydrodynamics (SPH).

2.4.2 Related Work

Numerical simulation of the Navier–Stokes equation has become a standard technique for the realistic animation of fluids. Foster and Metaxas [19] introduced a fully three-dimensional Navier–Stokes solver into computer graphics, and an effective and robust solution to the Navier–Stokes equation that includes semi-Lagrangian advection was reported by Stam [77]. Foster and Fedkiw [16] used a conjugate gradient method to solve the Poisson equation and an implicit level-set surface to represent the interface area effectively. Their method smooths the fluid interface, and changes of topology are represented robustly. This method has been extended to a particle level-set method by Enright et al. [14], with the addition of conservation of mass. Losasso et al. [47] used an adaptive octree data structure to show detailed fluid effects such as the crown phenomenon. Losasso et al. [51] then simulated various immiscible fluids such as oil, water, or fire using multiple level-sets. Hong and Kim [27] dealt with interface discontinuities using the GFM [15, 36]. They considered surface tension at the interface between two immiscible fluids at the projection step and introduced a discontinuous viscosity condition. Shin and Kim [73] modeled the force that drives liquids to a target shape using a pressure jump within the GFM. We model the viscous fingering phenomenon in a similar way.

There has been a spate of recent research in the computer graphics community, on the animation of phase transitions. Mihalef et al. [57] animated air bubbles in boiling water. They controlled the number and the volume of the bubbles produced by applying the equation of mass transfer by heat. Kim et al. [39] conserved the volume of air bubbles by revising the local value of divergence. Losasso et al. [49]

simulated phase transitions such as ice melting and paper burning. Wojtan et al. [86] animated corrosion and erosion of solid. In our scenario, phase changes, liquid to liquid as well, result from chemical reactions.

Desbrun and Gascuel [7] modified the SPH method to handle viscous fluids, and Müller et al. [59] proposed an interactive method in which SPH underpins the simulation of water; these authors also developed a multiphase SPH method to describe fluids of different compositions [60]. Cleary et al. [6] have improved the realism with which the collision of foam and bubbles on a complicated surface can be modeled.

Recently, much of effort has been applied to improving the efficiency of fluid simulations by modeling the details of fluid behavior on an underlying subgrid, which can be achieved by combining a Eulerian grid with Lagrangian particles. Kim et al. [35] modeled splashing by combining the SPH method with escaped particles from a particle level-set. Losasso et al. [52] developed a two-way coupling between a particle level-set and SPH to simulate diffuse regions such as splashing. Hong et al. [28] used SPH to model lively air bubbles on a coarse grid while retaining small-scale features of the flow. Lee et al. [48] proposed a way of making particle and level-set representations more interchangeable. We use the techniques mentioned above to simulate floating particles representing concentration factors.

2.4.3 Modeling Miscible Fluids with Multiple Level-Sets

To simulate fluids mixing, it is necessary to simulate more than two fluids and track the interfaces between them. Losasso et al. [51] tracked numerous interfaces of immiscible fluids using multiple level-sets, and this is the approach that we use here. We use multiple level-sets in miscible fluids to trace mixing surfaces in which we make a pressure jump according to Darcy's Law and simulate pressure term. We then trace the new mixing surfaces that viscous fingering creates. Chemical mass transfer also occurs at mixing surfaces.

We use fields containing the velocity of each substance for each multiple level-set. When using a single velocity field, it is difficult to represent the mixing surfaces' characteristics described above, because the velocity field of each fluid is scattered by the effect of the diffusion and pressure terms. Therefore, we must simulate scenarios that satisfy the divergence-free condition for each fluid while considering that fluid's changes of phase. We calculate the pressure term for each fluid incorporating the effect of Darcy's Law and mass transfer. Each velocity field is extrapolated and then advected by the semi-Lagrangian advection method. We create a single velocity field for the fluid by combining the calculated velocity fields and calculate the pressure term for the velocity field of the entire fluid in an additional step. The velocity field must be adjusted so that it fulfills the divergence-free condition, and finally we calculate the diffusion term. We perform this simulation repeatedly while dividing the velocity field to match the multiple level-sets. This method is expensive, but it is able to control the changes to the multiple level-sets that occur after reactions and interactions, so it is necessary. Figure 2.22 shows a schematic outline of our method.

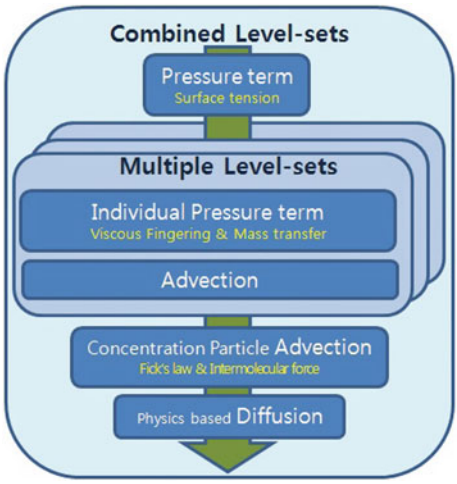

Fig. 2.22 Schematic outline of our simulation

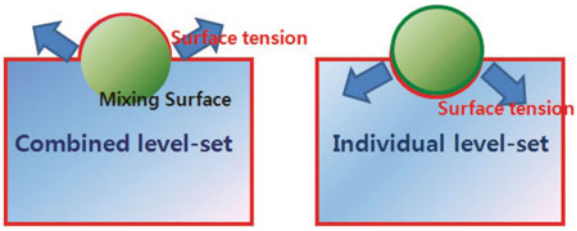

Fig. 2.23 When using a combined level-set, surface tension (*left*) is more accurate than when using individual level-sets (*right*)

In this section, we create and use combined level-sets to describe the mixing surfaces of entire fluids, as shown in Fig. 2.23. This method allows us to choose what data to assign to the combined level-set of the entire fluid, and to the level-set of each fluid when calculating surface tensions. When using numerous level-sets, it is possible for the direction of the surface tension to be miscalculated at the intersection between level-sets, as shown in Fig. 2.23 (right). We therefore consider all the level-sets to represent a single fluid, and calculate the surface tension using the combined level-set data. This allows us to determine the correct direction for the surface tension across the whole mixed fluid, as shown in Fig. 2.23 (left).

2.4.4 Basic Fluids Simulation

The Navier–Stokes equation, which is the basis of our simulation, preserves mass and momentum:

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\nabla \cdot \tau}{\rho} - \frac{\nabla p}{\rho} + \mathbf{f} \tag{2.61}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.62}$$

where $\mathbf{u}$ is the velocity, τ is the viscous stress tensor, and ρ is the density. The term $\mathbf{f}$ can be used to add external forces such as gravity and buoyancy. The numerical simulation of Eqs. (2.61) and (2.62) requires the value of $\mathbf{u}$ to be updated from $\mathbf{u}^n$ to $\mathbf{u}^{n+1}$ at the nth time-step. We discretize Eq. (2.61) by splitting it into two equations by introducing an intermediate velocity $\mathbf{u}^*$:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{\nabla \cdot \tau}{\rho} + \mathbf{f} \tag{2.63}$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{\nabla p}{\rho}. \tag{2.64}$$

The variable $\mathbf{u}^*$ can be used to compute the advection term using the semi-Lagrangian method of Stam [77]. We can write the divergence of Eq. (2.64) as a form of Poisson's equation:

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u}^*. \tag{2.65}$$

Once the pressure profile has been determined by solving this equation, we can obtain the final velocity profile:

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p. \tag{2.66}$$

There is a discontinuous pressure profile at the interface between two different fluids. It is possible to take the discontinuous pressure at the interface into account, using GFM [27, 36]. The pressure at node i, which is p_i, and the pressure at node $i+1$, which is p_{i+1}, are extrapolated across Γ to determine the ghost values, p_{i+1}^G and p_i^G :

$$p_i^G = p_i + J \tag{2.67}$$

$$p_{i+1}^G = p_{i+1} - J. \tag{2.68}$$

Using these equations, Eq. (2.65) can be expanded as follows [27]:

$$\frac{p_{i+1} + p_{i-1} - 2p_i}{\Delta x^2} = D(x_i) + \frac{J}{\Delta x^2} \tag{2.69}$$

$$\frac{p_{i+2} + p_i - 2p_{i+1}}{\Delta x^2} = D(x_{i+1}) - \frac{J}{\Delta x^2}, \tag{2.70}$$

where D represents the right-hand side of Eq. (2.65) in one dimension. These equations can be solved using the linear system that has been used to solve Poisson's equation.

Generally, pressure jumps are modeled by surface tension, but we model ∇P^{Darcy} as the pressure jump J generated at the interface of mixing fluids. This J causes viscous fingering, and will be explained in Sect. 2.4.5.

We use an octree in the same way as Losasso et al. [47] to focus on the fluid interface, and can create a smooth surface from a complicated liquid interface by means of the particle level-set method [14].

2.4.5 Viscous Fingering

When fluids mix, we can see them spread out irregularly as their mixing surface makes a fractal-like shape. Viscous fingering refers to the onset and evolution of these instabilities in the displacement of fluids. The unstable flow of a fluid in a porous medium, or by analogy in a Hele-Shaw cell, has been studied for 50 years [30]. The results have applications in areas such as enhanced oil recovery and microfluidics.

In our scenario, we model viscous fingering with Darcy's Law, which expresses the state of a fluid passing through a porous medium. We assume that a mixing process is equivalent to a fluid infiltrating a fluid consisting of molecules. At the mixing surface between two fluids, there is a pressure jump that results from the properties of the fluid. According to Darcy's Law, the pressure gradient vector is:

$$\nabla P^{Darcy} = \frac{\hat{\mu}\mathbf{U}}{k} + \hat{\rho}g, \tag{2.71}$$

where $\hat{\mu}$ is viscosity, $\mathbf{U}$ is velocity, k is permeability, $\hat{\rho}$ is density, and g is gravity.

There is a discontinuity of density and viscosity at an interface between two fluids. To calculate the density at a fluid interface exactly, we use an equation proposed by Losasso et al. [51]:

$$\hat{\beta} = (\beta^{-}\beta^{+})/\{\theta\beta^{+} + (1-\theta)\beta^{-}\}, \tag{2.72}$$

where $\theta = |\phi(x_i)|/(|\phi(x_i)|+|\phi(x_{i+1})|)$, and the $-$ and $+$ superscripts refer to values from different sides of the interface. The viscosity at the interface can be calculated in a similar way. As mentioned in Sect. 2.4.3, each substance has its own velocity field, and $\mathbf{U}$ is the velocity of the solute.

The porosity equation is formulated using the curvature of the interface of the solute that is infiltrating into a fluid. As shown in Fig. 2.24, if the curvature is high, the fluid surface is convex and the probability of infiltration is high, but if this shape is flat or concave, the probability of infiltration is low. Thus, we express porosity as $k = \alpha\kappa_{levelset}$, where α is a constant that we set to 0.1, and $\kappa_{levelset}$ is the curvature of the level-set, which can be expressed as follows: $\kappa_{levelset} = \nabla \cdot \left(\frac{\nabla\phi}{||\nabla\phi||}\right)$.

We define ∇P^{Darcy} as the pressure jump J generated at the interface of mixing fluids. The u, v, and w velocities and pressure jump J are defined at the centers of each face of a cell and referenced locally [19]. The pressure jump is therefore

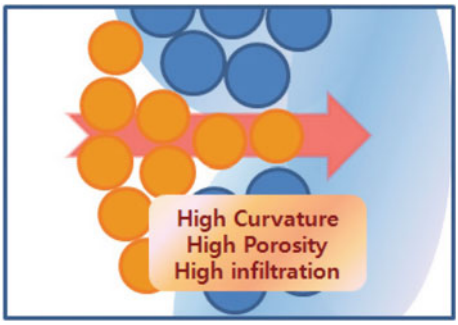

Fig. 2.24 In our model, if the curvature is high, the porosity is high and the probability of infiltration is high

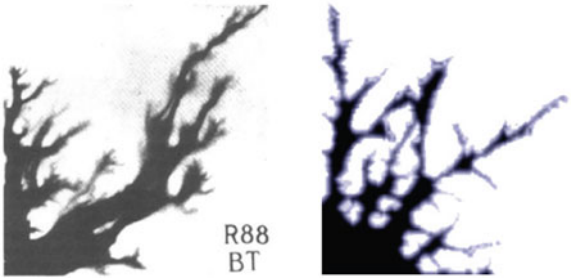

Fig. 2.25 Viscous fingering. Miscible fluids mixing (*left*), and our simulation result (*right*)

calculated on the face of the cell that is defined as the interface in which the neighbor cell has opposite sign. Therefore, **U** is used for the velocity of the cell face, which is a scalar. For example, if there are interfaces on three cell faces, individually calculated J value for each cell face are stored. J consists of a scalar value for each dimension (x, y, z):

$$J_{(x,y,z)} = \frac{\hat{\mu}\mathbf{U}_{(x,y,z)}}{k} + \hat{\rho}\mathbf{g}_{(z)}. \tag{2.73}$$

This formulation allows us to demonstrate viscous fingering at a mixing surface, as shown in 2D in Fig. 2.25. This is a similar structure to that shown in a real photograph of mixing fluids, due to Habermann [30], also reproduced in Fig. 2.25.

2.4.6 Chemical Mass Transfer

When fluids mix, chemical reactions can occur. Then, the volume of fluid may decrease or increase and some of the fluid may change into another type of fluid. We model and simulate this kind of mass transfer. Mihalef et al. [57] simulated the mass transfer that occurs when water boils. The mass transfer because of chemical reaction is similar to the phase transition that can be caused by heat. The rate of mass transfer resulting from a chemical reaction can be expressed as follows:

$$\dot{m}_c = -\frac{\mathbf{cf}_{solute} \cdot \mathbf{N} - \mathbf{cf}_{solvent} \cdot \mathbf{N}}{H}, \tag{2.74}$$

where $\mathbf{cf}$ is the chemical flux defined by $\mathbf{cf} = -D\nabla C_{cell}$, $\mathbf{N}$ is the outward facing normal from $\mathbf{cf}_{solute}$ to $\mathbf{cf}_{solvent}$, H is the heat of reaction, C_{cell} is the concentration of each cell, and D is the coefficient of diffusivity for the concentration.

The rate of mass transfer by chemical reaction depends on the concentration of the substance. Depending on the rate of mass transfer, the level-set of the fluid is updated as follows:

$$\phi_* = \phi_n - \Delta t \frac{\dot{m}_c}{\hat{\rho}} |\nabla \phi|. \tag{2.75}$$

ϕ_* is advected to ϕ_{n+1} using level-set method after updated. Because the volume of the fluid changes, we must revise the divergence value of the pressure term by using the rate of mass transfer to control the volume of fluid, as follows:

$$\nabla \cdot \mathbf{u}^{n+1} = \frac{\dot{m}_c}{\hat{\rho}} |\mathbf{N}|, \tag{2.76}$$

$$\nabla^2 p = \frac{\hat{\rho}}{\Delta t} \nabla \cdot \mathbf{u}^* - \frac{\dot{m}_c}{\Delta t} |\mathbf{N}|. \tag{2.77}$$

This method is similar to those explained in [39, 57]. This allows us to simulate phenomena in which the volume of the substance changes, such as the melting of a solid because of a chemical reaction. Figure 2.26 shows a solid teapot melting in a transparent liquid. Because of mass transfer, the volume of the teapot decreases. The solid teapot shape is defined by an implicit surface using level-set data, which makes it easy to implement this scenario.

2.4.7 Hybrid Method

It is hard to model intermolecular forces using only grid-based advection of concentration. We therefore simulate the effect of molecular diffusion and interaction on concentration using what we call *concentration particles*. The advection of concentration is simulated using both a grid-based model of semi-Lagrangian advection and particle-based advection. We can assume that the concentration particles are not completely absorbed in the solvent in which they dissolve, but float about like oversize molecules. Figure 2.27 shows a 2D simulation of the mixing of two substances using this hybrid method.

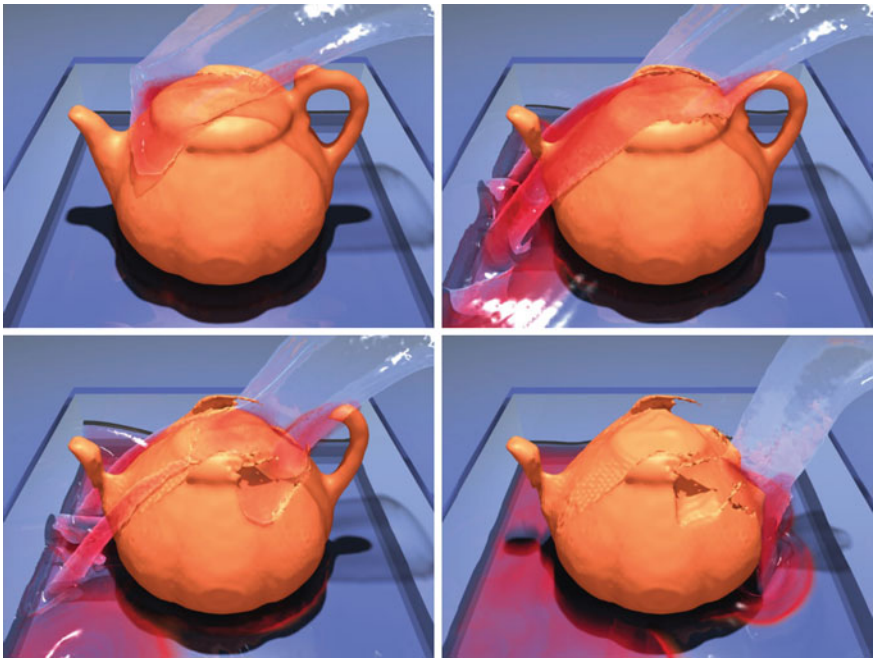

Fig. 2.26 Pouring water on the teapot. The solid teapot melts by chemical mass transfer. (grids:128 × 128 × 128; particles: about 20,000)

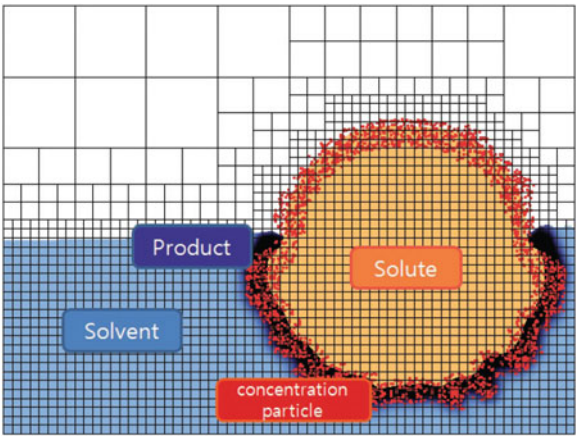

Fig. 2.27 Hybrid simulation of particle and grid-based method. The grid is 64 × 64 (6-level octree). The *blue* and *yellow* points are the level-sets and the *red* points are concentration particles. The *dark blue* points are particles modeling the concentration after the reaction

2.4.7.1 Concentration Particles and Absorption

Concentration particles are generated in the grid cells corresponding to a mixing surface. They are defined by position, velocity, radius, and concentration. The number of particles that are reseeded depends on the maximum that has been set for each

cell. That is, we simulate more particles with reseeding to have max_p particles in a mixing surface cell if the number of particles is less than max_p. max_p is the maximum number of particles that a cell can have. In our experiment, we set max_p to four or eight. When a particle is seeded, its concentration is initialized to $C_{equilibrium}/max_p$.

When one or more concentration particles exist in a cell, absorption occurs from the particles to the grid. The reaction rate is proportional to the concentration of the substance. In general, the reaction rate of two substances is $k_{reaction}[A][B]$, where $[A]$ is the sum of the particle concentrations existing in each cell ($\sum_i^{np} C^n_{particle_i}$) and $[B]$ is the concentration of solvent in the cell.

In the simulation, we limit the maximum number of cell concentrations to $C_{equilibrium}$, and we assume that the concentration of solvent $[B]$ is initialized to $C_{equilibrium}$. Because the concentration of substance $[B]$ is proportionally reduced as reaction product is generated, we can model $[B]$ as $(C_{equilibrium} - C_{cell})$. Thus, the cell concentration of reaction product absorbed from a particle for the next time-step, C^{n+1}_{cell} is:

$$C^{n+1}_{cell} = C^n_{cell} + \sum_i^{np} k_{reaction} C^n_{particle_i} (C_{equilibrium} - C^n_{cell}), \tag{2.78}$$

where C_{cell} is the concentration of product in the cell, $C_{particle_i}$ is the ith particle's concentration in the cell, np is the number of particles in the cell, and $k_{reaction}$ is a coefficient that defines the rate of the chemical reaction. In our experiment, we use a small number, typically 0.001, for $k_{reaction}$ and set $C_{equilibrium}$ to 1.0.

When a concentration particle is absorbed into the cell, the particle concentration of the solute is reduced:

$$C^{n+1}_{particle} = C^n_{particle} - k_{reaction} C^n_{particle} (C_{equilibrium} - C^n_{cell}). \tag{2.79}$$

Depending on Eq. (2.79), if a particle's concentration is less than a fixed threshold, which is 0.01, the particle is deleted. If C^n_{cell} is larger than $C^n_{equilibrium}$, the concentration of product decreases and the particle concentration of solute increases in the next step. The process is similar to maintaining a state of chemical equilibrium. This allows the computation to be simplified without a significant effect on accuracy.

2.4.7.2 Concentration Particle Advection

Concentration particles are advected by their own velocity according to a Lagrangian method. Each of their positions is updated using the equation: $\mathbf{p}^{n+1}_{particle} = \mathbf{p}^n_{particle} + \mathbf{u}^n_{particle} \cdot \Delta t$. The velocity of a particle $\mathbf{u}_{particle}$ is calculated using the equation: $\mathbf{u}^{n+1}_{particle} = \mathbf{u}^n_{particle} + \mathbf{f}_{\text{advection}} \cdot \Delta t$. The advection of concentration particles depends on the intermolecular diffusion, infiltration, and coupling forces:

$$\mathbf{f}_{\text{advection}} = \mathbf{f}_{\text{diffusion}} + \mathbf{f}_{\text{capillary}} + \mathbf{f}_{\text{molecular}} + \mathbf{f}_{\text{coupling}}. \tag{2.80}$$

Concentration particles spread out in the direction of the concentration gradient. The diffusive flux follows Fick's Law:

$$\mathbf{f}_{\text{diffusion}} = -D\nabla C_{cell}, \tag{2.81}$$

where D is the diffusion coefficient. It is not easy to model complicated diffusion scenarios in which irregular filaments are produced and spread out, using only the above Eq. (2.81). Thus, we use a capillary force and intermolecular forces.

The capillary force is modeled using the gradient of curvature of the concentration field. Because regions of high curvature have a high porosity, it is easy for a fluid to infiltrate in that region. We therefore model the capillary force as follows:

$$\mathbf{f}_{\text{capillary}} = I|\kappa_{concentration}|\nabla\kappa_{concentration}, \tag{2.82}$$

where $\kappa_{concentration}$ is the curvature of the concentration field and I is the infiltration coefficient. The value of $\kappa_{concentration}$ can be expressed as $\kappa_{concentration} = \nabla \cdot (\nabla C_{cell}/||\nabla C_{cell}||)$.

We simulate intermolecular forces using the SPH method. The intermolecular forces themselves are modeled by the interaction forces between concentration particles. The following equation, which is typically used to calculate pressure in SPH-based fluid simulations, has been found experimentally to be effective in modeling the attractive force between concentration particles:

$$\mathbf{f}_{\text{molecular}} = A\sum_j m_j \left(\frac{P_i}{{C_{particle_i}}^2} + \frac{P_j}{{C_{particle_j}}^2} \right) \nabla W(\mathbf{x}_j - \mathbf{x}_i), \tag{2.83}$$

where the pressure $P_i = \alpha C_{particle_i}$ with the control parameter α, A is the attraction coefficient, and m is the mass of a particle. We assume that the mass of all the particles is the same, with a value of 1.

When a concentration particle moves between different fluids, a resistance force is provided by the fluid into which the particle is moving. This force causes the coupling between the velocity of grid and particle. The magnitude of this force is proportional to the relative velocity of the concentration particle and the receiving fluid, and can be expressed as:

$$\mathbf{f}_{\text{coupling}} = -R(\mathbf{u}_{particle} - \mathbf{u}_{cell}), \tag{2.84}$$

where R is the resistance coefficient. The overall advection of the particles is provided by the sum of the forces in Eqs. (2.81)–(2.84).

2.4.8 Results

Simulations were performed on an Intel PC with a 3.0 GHz CPU. In general, simulation grids are economically implemented as octrees, but we must use a regular grid for scenarios where concentration must be modeled. In this case, the value of the level-set is less than 0. In other words, we can simulate concentrations where liquid is present without ambient air. Therefore, if ϕ over the whole level-set is more than 0, we could use an octree data structure instead of the dense regular grid that we used where ϕ is less than 0.

Because an octree data structure is being used, the computation cost increases in proportion to the volume of fluid that is simulated. The simulation shown in Fig. 2.26 takes at least 15 s per frame and at most 120 s. Mental Ray in Maya is used for rendering. Simulation data are saved as a mesh and a scalar field to be rendered in Maya.

Figure 2.28 shows that the Venus-shaped volume of fluid dives into an open water surface and is mixed. The two different fluids intermingle and spread out just as ink mixes with water. This experiment was performed using a $256 \times 256 \times 128$ grid. The simulations took approximately 9 min per frame, including running the fluid solver and the file I/O.

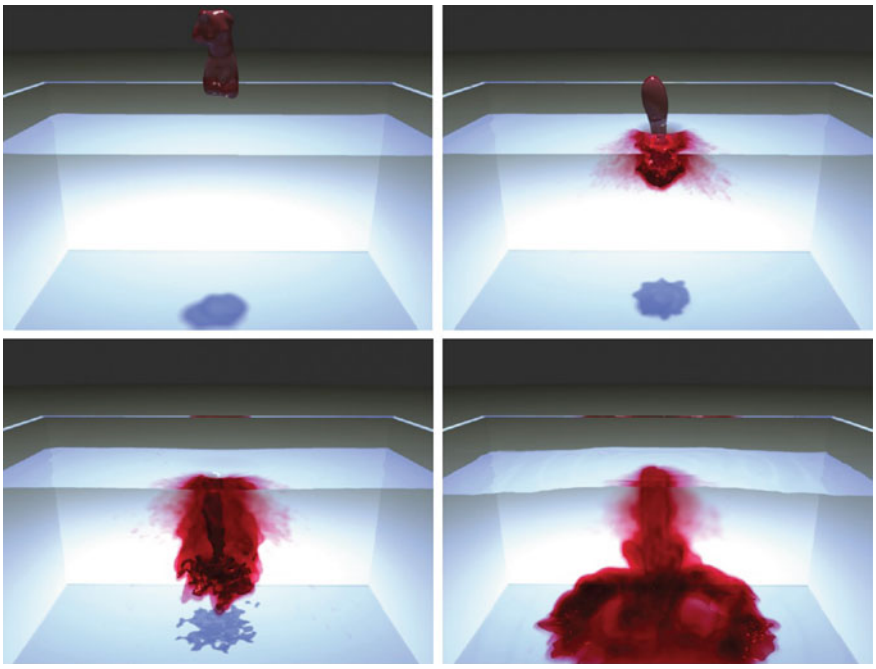

Fig. 2.28 Dropping *red ink*. Two different liquids mix (grids: $256 \times 256 \times 128$, particles: about 120,000)

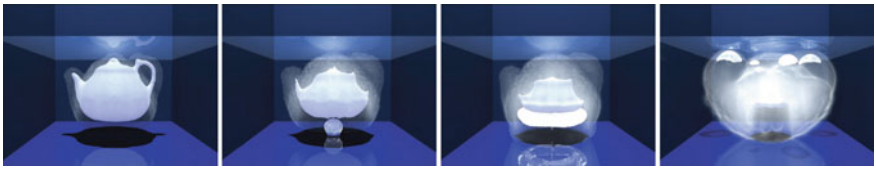

Fig. 2.29 Dissolving a solid teapot. By mass transfer, the volume of the solid decreases (grids: $128 \times 128 \times 128$, particles: about 30,000)

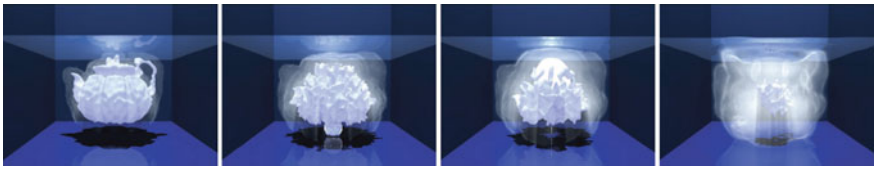

Fig. 2.30 Dissolving a liquid teapot. Viscous fingering occurs at the mixing surface (grids, particles: same as above)

Figure 2.29 shows how a solid teapot shape dissolves in a liquid. As the volume of the solid is reduced by mass transfer, the solid teapot dissolves in water. Violet material from the teapot that has been mixed in water floats on the surface and interacts with air bubbles. Figure 2.30 shows a somewhat different scenario in which a liquid teapot mixes with the water that surrounds it. In contrast to Fig. 2.29, viscous fingering now takes place because of the difference in pressure at the interface between the two fluids and a liquid core itself flows. Bubbles were intentionally inserted to show the comparison between the solid and the liquid when interacting with bubbles. We can thus see the complicated phenomena of liquids mixing. The size of grid for the simulations shown in Figs. 2.29 and 2.30 was 128^3. The average simulation time per frame was 120 s.

Figure 2.31 presents a dramatic scenario in which a liquid teapot and a rigid teapot are dropped into water one by one. First, it shows how they react when they are of the same type of substance, clear water. Second, the insoluble object drops, and when it reaches the bottom of the water, it turns into blue ink teapot. It slowly dissolves and mixes with the water. The grid for this simulation was 128^3, and the average simulation time per frame was 100 s.

2.4.9 Conclusions

This section has described a technique for modeling the flow of miscible multiphase fluids by improving the handling of interfacial properties and chemical reactions. In several experiments, we constructed naturalistic scenarios in which a solid body melts or liquids are mixed. These combinations of viscous fingering, chemical-based mass

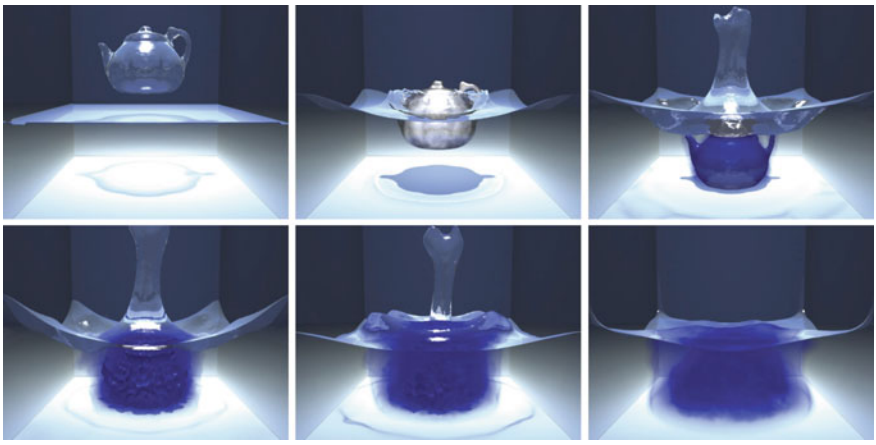

Fig. 2.31 Water meets three types of teapots: clear water, steel, and *blue ink* (grid:$128 \times 128 \times 128$, particles: about 25,000)

transfer, and molecular forces are relatively easy to model with techniques familiar to the computer graphics community. In the future, it can be researched to simulate smaller scale features such as the filaments that appear when ink spreads out in water. And also, simulations for the liquid–solid reaction could be performed, based on accurate chemical laws such as solution and solidification with a state of chemical equilibrium.

2.5 Anisotropic Particle Level-Set Method for Multiphase Fluid

Abstract This section presents how to track the surface of a multiphase fluid more accurately by using the particle level-set method with anisotropic particles instead of spherical particles. While we use the weighted version of principal component analysis (WPCA) to construct the anisotropic particles, its computational cost is high. We adopt the distribution of particles from the directional derivative to generate the anisotropic particles. Compared to particle level-set method, this approach provides more details of surface, corrects numerical dissipation, and preserves the volume of the fluid. Furthermore, this section presents particle-based fluid simulations with surface reconstruction that uses anisotropic particles.

2.5.1 Introduction

Accurately capturing the interfaces of a multiphase fluid is a challenging problem in computer graphics. The level-set method within a Eulerian simulation can track the free surface of a liquid [16]. However, the standard grid-based semi-Lagrangian advection method only has first-order accuracy, so there is a large amount of numerical dissipation. Attempts to track the fluid interface more accurately have included the use of various triangle meshes or marker particles. Mesh-based surface tracking combines existing resampling methods with the use of convex hulls to connect surface features during topological changes [83, 87]. The particle level-set method corrects the level-set near the surface. Enright et al. [14] added Lagrangian particles to the level-set in order to represent surface details more accurately. To implement this method, marker particles are seeded near the surface and advected by a velocity which is tri-linearly interpolated at the particle position. Then the particles can be used to correct errors caused by numerical dissipation. To calculate the level-set on both sides of the fluid interface, the particles are considered as spheres, with radii determined by their distance from the surface. This method is widely used because it is very simple and resulting surface preserves the volume of the fluid.

This section proposes an anisotropic particle level-set method to achieve a more accurate fluid interface than the spherical particles. We create an anisotropic particle by considering distribution of particles. Though use of a weighted version of principal component analysis(WPCA) results the fluid surface accurately, the search for neighboring particles required to calculating singular value decomposition involves a high computational cost. So this section devises a new method in which directional derivative is used to generate the anisotropic particles. This reduces the computational costs but still allows the fluid interface to be tracked more accurately.

2.5.2 Related Work

The simulation of liquid using the Navier–Stokes equation has been researched for a long time. Foster and Metaxas [19] developed a three-dimensional Navier–Stokes method for fluid simulation. Stam [77] proposed a semi-Lagrangian integration scheme to simulate unconditionally stable fluids using a three-dimensional grid. Losasso et al. [47] utilized an adaptive octree structure to obtain a high resolution surface. Hong and Kim [27] considered surface tension between multiphase fluids using the ghost fluid method (GFM) to deal with discontinuities at the fluid interface. To represent the fluid surface more accurately, Enright et al. [14] proposed the particle level-set method. Then, they improved this method by introducing a semi-Lagrangian approach to track the surface more accurately and rapidly. Mihalef et al. [56] handled the dynamics of a liquid and its surface color. Ianniell and Mascio [33] tracked the interfaces by Lagrangian oriented particles in conjunction with a level-set. Advection of simulation is also for greater accuracy, while CIP [76] ensure

high order accuracy. Furthermore, there are several hybrid approaches for bubble [22] and splash [28, 35]. Losasso et al. [52] used two-way coupled particle level-set and SPH [48] in order to simulate diffuse regions such as splashing. Kim et al. [38] simulate an ellipsoidal shape of an air bubble by a drag force on its upper surface. Wojtan et al. [88] were able to represent detailed fluid surfaces with thinner features using triangle meshes. Our anisotropic particle level-set approach has been inspired by anisotropic particle methods. Liu et al. [50] employed anisotropic kernels in the SPH simulation. Yu and Turk [90] formulated anisotropic smoothing kernels using the WPCA method. Jo et al. [34] simulated SPH-based fluids using anisotropic kernels formed by the particle velocities. We refer to works of Donia et al. [8], Zheng et al. [92], and Rahman and Murshed [68] for creation of anisotropic kernel. Donia et al. [8] propose a texture generation method by computing their movement of a motion distribution followed by the generation of image frames. Zheng et al. [92] detected a pattern by judging of eclipse and Support Vector Machines (SVM). Other interesting works include viscoelastic fluids [21], control methodology [83], vortex particle [75], and subgrid turbulence model [70].

2.5.3 Particle Level-Set Method (PLS)

We use the Navier–Stokes equation to simulate large volumes of liquid. The momentum conservation equation is

$$\mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\nabla p}{\rho} = \mathbf{f} \tag{2.85}$$

and the mass conservation equation is

$$\nabla \cdot \mathbf{u} = 0 \tag{2.86}$$

where $\mathbf{u} = (u, v, w)$ is velocity, p is pressure, and $\mathbf{f}$ is the sum of the external forces, including gravity and control forces. We use octree structures [47] for fast grid simulation, back and forth error compensation and correction (BFECC) [40] to achieve second-order accuracy in fluid volume preservation, and the particle level-set method [14] to represent complicated fluid surfaces.

First, marker particles are seeded in the surface region. The radius r_p of a particle as follows:

$$r_p = \begin{cases} r_{max} & \text{if } s_p\phi(\mathbf{x}_p) > r_{max} \\ s_p\phi(\mathbf{x}_p) & \text{if } r_{min} \le s_p\phi(\mathbf{x}_p) \le r_{max} \\ r_{min} & \text{if } s_p\phi(\mathbf{x}_p) < r_{min} \end{cases} \tag{2.87}$$

where s_p is the sign of the particle, $\phi(\mathbf{x}_p)$ is the implicit function, and $\mathbf{x}_p$ is the particle position. The minimum radius is $0.1 \cdot \min(dx, dy, dz)$ and the maximum radius is

$0.5 \cdot \min(dx, dy, dz)$. In most of the examples presented later in this paper, 32 marker particles are used each cell. Then the level-set is integrated using Eq. (2.85), while the particles are advected with the interpolated velocity at their positions. The error correction scheme proposed by Enright et al. [14] uses spherical particles. A spherical implicit function $\phi_p(\mathbf{x})$ is determined by the marker particle radius, as follows:

$$\phi_p(\mathbf{x}) = s_p(r_p - |\mathbf{x} - \mathbf{x}_p|) \tag{2.88}$$

where r_p is the particle radius and $\mathbf{x}$ is the position of the node. Then a new value of the level-set is determined by comparing the spherical implicit function $\phi_p(\mathbf{x})$ with the current level-set value. The positive and negative level-set values are then obtained as follows:

$$\phi^+ = \max_{\forall p \in \mathbf{E}^+}(\phi_p, \phi^+) \tag{2.89}$$

$$\phi^- = \max_{\forall p \in \mathbf{E}^-}(\phi_p, \phi^-) \tag{2.90}$$

where ϕ^+ is the level-set value in the $\phi > 0$ region, ϕ^- is the level-set value in the $\phi < 0$ region, $\mathbf{E}^+$ is the set of escaped positive particles, and $\mathbf{E}^-$ is the set of escaped negative particles. More details on this are given elsewhere [13, 14].

2.5.4 Anisotropic Particle Level-Set Method (APLS)

We use anisotropic, instead of spherical, particles in the particle level-set method. We specify an anisotropic particle by its position, its three axes and its magnitudes along three axes.

2.5.4.1 Determining Particle Axes and Magnitudes Using WPCA

We first determine the axes of a particle from the distribution of neighboring particles by employing the WPCA Method [90]. We compute weighted mean $\mathbf{x}_i^w$ from the neighboring particle to construct the covariance matrix, as follows:

$$\mathbf{x}_i^w = \frac{\sum_j w_{ij}\mathbf{x}_j}{\sum_j w_{ij}} \min_{\forall p \in \mathbf{E}^-}(\phi_p, \phi^-) \tag{2.91}$$

The weight function w_{ij} is calculated by using neighbor particles which have the same sign within radius r_i, as follows:

$$w_{ij} = \begin{cases} 1 - (||\mathbf{x}_i - \mathbf{x}_j||)^3 & \text{if} \quad ||\mathbf{x}_i - \mathbf{x}_j|| < r_i \\ 0 & \text{otherwise} \end{cases} \tag{2.92}$$

Then, the covariance matrix $\mathbf{C}_i$ is obtained as follows:

$$\mathbf{C}_i = \frac{\sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i^w)(\mathbf{x}_j - \mathbf{x}_i^w)^T}{\sum_j w_{ij}} \tag{2.93}$$

For each marker particle, we perform a singular value decomposition of the covariance matrix $\mathbf{C}_i$ to obtain the eigenvectors and eigenvalues, which become the axes and magnitudes ($\sigma_1 < \sigma_2 < \sigma_3$) of the anisotropic particle.

2.5.4.2 Error Correction Using Anisotropic Particles

Because we use anisotropic particles, we need to modify the error corrections scheme of the particle level-set method. We can obtain the radius of an anisotropic particle r_p in any direction by solving Eq. (2.94). The local coordinate system of each particle transforms the position of the node. The local coordinate system is determined by three axes in Sect. 2.5.4.1. We can calculate r_p for an arbitrary direction as follows:

$$r_p = \sqrt{\frac{1}{\frac{x_{lx}^2}{\sigma_1'^2} + \frac{x_{ly}^2}{\sigma_2'^2} + \frac{x_{lz}^2}{\sigma_3'^2}}} \tag{2.94}$$

where $\mathbf{x}_l = (x_{lx}, x_{ly}, x_{lz})$ is the position of node in the local coordinate system, σ_1' is the shortest distance (ϕ_p) to the surface, $\sigma_2' = (\sigma_2/\sigma_1)\phi_p$ and $\sigma_3' = (\sigma_3/\sigma_1)\phi_p$. So we apply this r_p to Eq. (2.88) and obtain the new $\phi_p(\mathbf{x})$ using anisotropic particle. This approach improves the representation of the level-set surface, but it takes about 21 times longer than particle level-set method. This is because the data structures used in the anisotropic particle level-set method using WPCA are inappropriate for neighbor searching. We could use fewer anisotropic particles but then the resulting surface is to be worse. Section 2.5.4.3 introduces a new method to avoid this problem.

2.5.4.3 Reducing of Computational Cost Using the Directional Derivative

If the number of anisotropic particles in APLS using WPCA decreases, the computational cost also decreases but the results would be of lower quality. Therefore, this section proposes a new method that has good performance in time consumption as well as improves representation of the surface. We generate an anisotropic particle using directional derivative instead of WPCA method. First, we discard the WPCA calculation. Second, we update the level-set using PLS. Third, we determine three axes of the anisotropic particle using directional derivative; the error correction module is computed in the anisotropic particle level-set method. More details are described in Algorithm 3.1.

Algorithm 3.1 APLS Method

```
for all leaf nodes do
  (re)seed marker particles and set their radius
for all leaf nodes do
  for all particles do
    Time integration of marker particles
for all leaf nodes do
  Time integration of the implicit function
for all leaf nodes do
  for all particles do
    delete and add marker particles
    compute updated φ_p(x) using PLS
for all leaf nodes do
  for all particles do
    compute normal at the its position (∇φ)
    set minor axis (e_1)
    set major axis e_2 using the directional derivative
    set last axis e_3= cross product(e_1, e_2)
for all leaf nodes do
  for all particles do
    compute new φ_p(x) using the anisotropic particles
```

The minor axis $\mathbf{e}_1$ is determined from the gradient of ϕ. The major axis is the axis with the minimum variation of level-set value. The direction of minimum variation is calculated using the directional derivative, as follows:

$$\nabla_{\mathbf{e}}\phi(\mathbf{x}) = \frac{\phi(\mathbf{x}+h\mathbf{e}) - \phi(\mathbf{x}-h\mathbf{e})}{2h} \tag{2.95}$$

where $\phi(\mathbf{x})$ is the level-set value at the position $\mathbf{x}$ and $\mathbf{e}$ is the direction that we seek, $||\mathbf{e}|| = 1$. In our example, h is the particle radius. To find the minimum variation, we calculate the directional derivative iteratively using Eq. (2.95) on the plane, normal to $\mathbf{e}_1$. The direction that has the minimum variation is the major axis $\mathbf{e}_2$. The $\mathbf{e}_3$ axis is calculated by the cross product of the minor axis $\mathbf{e}_1$ and the major axis $\mathbf{e}_2$. We can introduce into Eq. (2.94) since we can obtain σ_1, σ_2 and σ_3 by calculating the directional derivative along the axes $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ using Eq. (2.95). The value of σ_1, σ_2, and σ_3 are inversely proportional to the variation of the level-set in the corresponding direction. This simple calculation improves the performance of the anisotropic particle level-set method using directional derivative.

2.5.4.4 Additional Trials

Our new method was applied to the surface reconstruction module of a particle-based fluid simulation using SPH [1, 59, 60]. Our simulations and surface reconstruction algorithms largely follow [90]. We only modify their method of setting the anisotropic kernels using WPCA.

The normal vector at each particle that is used to calculating the surface tension force is the minor axis of the anisotropic particle. We calculate the rate of density change for all neighboring particles. If a particle j has the smallest rate of density change, we temporarily align the vector $\mathbf{x}_j - \mathbf{x}_i$ with the major axis. The direction of the third axis is the cross product of the minor and major axes. However, the major and minor axes must be at right align to each other. So we have to change the direction of the major axis to become the cross product of the direction of the minor axis and the third axis. The value of σ_1, σ_2 and σ_3 are inversely proportional to the rate of density change along the corresponding axes. This approach eliminates the computational time required to obtain three eigenvectors and three eigenvalues.

	Algorithm 3.2.1 Particle-based Fluid Simulation
1	**for all** particles $i \in \mathbf{S}$ **do**
2	find neighbors $\mathbf{N}_i$
3	**for all** particles $i \in \mathbf{S}$ **do**
4	compute ρ_i, p_i
5	**for all** particles $i \in \mathbf{S}$ **do**
6	compute forces (gravity, pressure, viscosity, surface tension)
7	**for all** particles $i \in \mathbf{S}$ (for all about previous step) **do**
8	compute anisotropic kernel (compute three axes) $\rightarrow$ Algorithm 3.2.2
9	compute node density (for surface reconstruction)
10	**for all** particles $i \in \mathbf{S}$ **do**
11	compute new $\mathbf{v}_i$, $\mathbf{x}_i$

	Algorithm 3.2.2 Compute Anisotropic Particle
1	major axis = normal vector of particles
2	**for all** particles $i \in \mathbf{N}_i$ **do**
3	compute rate of density change i, j ($\Delta\rho_{ij}$)
4	**if** ($\Delta\rho_{min} > \Delta\rho_{ij}$)
5	$\Delta\rho_{min} = \Delta\rho_{ij}$
6	major axis = $\mathbf{x}_j - \mathbf{x}_i$
7	**end if**
8	the other axis = cross product(minor axis, major axis)
9	major axis = cross product(minor axis, the other axis)

2.5.5 Results

Simulations were performed on an Intel Core i7 CPU running at 2.93 GHz, and rendered the fluid models by ray tracing. BFECC (back and forth error compensation and correction) was used for the advection, with an octree grid. The maximum depth

of the octree is 7, so yielding a $128 \times 128 \times 128$ grid. Monte Carlo integration was used to measure the volume of the fluid.

Figure 2.32 shows that the water ball bounced back from the surface. Figure 2.32a is the 21st frame with the PLS. It takes 0.7226 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00129. Figure 2.32b is the 21st frame with the APLS using WPCA. It takes 15.8472 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00123. Figure 2.32c is the 21st frame with the APLS using directional derivative. It takes 0.7477 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00107. Figure 2.32c shows complex water surface by adding trivial time.

Figure 2.33 shows that the water ball was dropped onto the fluid surface. Figure 2.33a is initial water ball drop and Fig. 2.33b is the 7th frame with the PLS. It takes 0.6172 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00027. Figure 2.33c is the 7th frame with the APLS using WPCA. It takes 14.3663 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00011. Figure 2.33d is the 7th frame with the APLS using directional derivative. It takes 0.6226 s to run the advection module with 32 particles, and the volume of fluid decreases by 0.00005. Figure 2.33d is similar to Fig. 2.33c for the volume, although the computational time of Fig. 2.33d is about 0.04333 times lower than the APLS with WPCA.

Fig. 2.32 Water drop tower: **a** PLS **b** APLS with WPCA **c** APLS with directional derivative

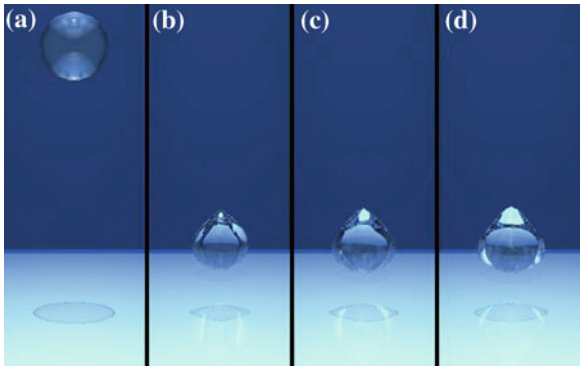

Fig. 2.33 Water ball drop: **a** The initial water ball drop **b** PLS **c** APLS with WPCA **d** APLS with directional derivative

Fig. 2.34 Water pouring into a tank: **a** 26th frame, **b** 43rd frame, **c** 60th frame

Table 2.1 Volume and time-consuming for water ball drop

Frame	PLS	APLS with WPCA	APLS with directional derivative
1st	1.15089 (0.6010 s)	1.15089 (14.0895 s)	1.15089 (0.6073 s)
7th	1.15062 (0.6172 s)	1.15078 (14.3663 s)	1.15084 (0.6226 s)
21st	1.1496 (0.7226 s)	1.14966 (15.8472 s)	1.14982 (0.7477 s)
70th	1.13582 (0.8449 s)	1.15089 (18.5929 s)	1.14146 (0.8896 s)
100th	1.12717 (0.0160 s)	1.15089 (22.6637 s)	1.13641 (1.0378 s)

Figure 2.34 shows water pouring into a tank. The stretching feature of the liquid's surface is observed in this physical behavior and visual result. Figure 2.34a shows a water source and it was added in every frame. Figure 2.34b shows water colliding with a wall. The anisotropic particle level-set helps the water surface maintain sharp features. Next, Fig. 2.34c shows the complex surface of the water.

Table 2.1 shows how the volume of fluid in Figs. 2.32 and 2.33 is conserved. Compared to the first frame, the large amount of volume with the PLS in the 70th frame and the 100th frame is lost. But the volume of the fluid is conserved in the third and fourth column. In the 100th frame, it takes 1.0160 s to run the advection module with PLS and 1.0378 s to run the advection module with APLS using directional derivative. As a result, the PLS and the APLS with directional derivative have the similar time-consuming, but the volume of fluid with APLS using directional derivative is more conserved.

2.5.6 Conclusion and Future Work

This section presented the anisotropic particle level-set method that captures the interface of the multiphase fluid accurately. The anisotropic particle is created by particles' distribution. To get the three axes, the gradient of level-set value and directional derivative are used. As a result, this anisotropic particle level-set method provides more accurate simulation than spherical particle level-set method and faster than APLS with WPCA. However, there is still numerical dissipation into thin feature. Anisotropic escaped particles can be included to handle this problem. They act as splash where numerical dissipation occurred.

References

1. Adams B, Pauly M, Keiser R, Guibas LJ (2007) Adaptively sampled particle fluids. In: Proceedings of ACM SIGGRAPH 2007. ACM transactions on graphics (TOG), vol 26(3), pp 481–487, July 2007
2. Brackbill JU, Kothe DB, Zemach C (1992) A continuum method for modeling surface tension. J Comput Phys 100(2):335–354
3. Becker M, Teschner M (2007) Weakly compressible SPH for free surface flows. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 209–217
4. Chorin AJ (1997) A numerical method for solving incompressible viscous flow problems. J Comput Phys 135(2):118–125
5. Carlson M, Mucha PJ, Van Horn II RB Turk G (2002) Melting and flowing. In: ACM SIGGRAPH symposium on computer animation, pp 167–174
6. Cleary PW, Pyo SH, Prakash M, Koo BK (2007) Bubbling and frothing liquids. In: Proceedings of ACM SIGGRAPH 2007. ACM transactions on graphics (TOG), vol 26(3), pp 971–976, July 2007
7. Desbrun M, Gascuel M-P (1996) Smoothed particles: a new paradigm for animating highly deformable bodies. In: Computer animation and simulation'96. pp 61–76
8. Donia T, Gabriel A, Serban, Andreea M, Rada M (2009) Textual entailment as a directional relation. J Res Pract Inf Technol 41(1)
9. de Sousa F, Mangiavacchi N, Nonato L, Castelo A, Tome M, Ferreira V, Cuminato J, Mckee S (2004) A front-tracking/front-capturing method for the simulation of 3d multi-fluid flows with free-surfaces. J Comput Phys 198(2):469–499
10. Durikovic R (2001) Animation of soap bubble dynamics, cluster formation and collision. Computer Graphics Forum (Eurographics 2001 Proceedings), vol 20(3), pp 67–76
11. Dyke MV (1982) An album of fluid motion. The Parabolic Press, Stanford
12. Enright D, Fedkiw R, Ferziger J, Mitchell I (2002) A hybrid particle level set method for improved interface capturing. J Comput Phys 183(1):83–116
13. Enright D, Losasso F, Fedkiw R (2005) A fast and accurate semi-lagrangian particle level set method. Comput Struct 83(6–7):479–490
14. Enright D, Marschner S, Fedkiw R (2002) Animation and rendering of complex water surfaces. In: Proceedings of ACM SIGGRAPH 2002. ACM transactions on graphics (TOG), vol 21(3), pp 736–744, July 2002
15. Fedkiw R, Aslam T, Merriman B, Osher S (1999) A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). J Comput Phys 152(2):457–492
16. Foster N, Fedkiw R (2001) Practical animation of liquids. In: Proceedings of ACM SIGGRAPH, pp 23–30

17. Fournier A, Habibi, Poulin P (1998) Simulating the flow of liquid droplets. In: Graphics interface'98, June 1998
18. Fattal R, Lischinski D (2004) Target-driven smoke animation. In: Proceedings of Proceedings of ACM SIGGRAPH 2004. ACM transactions on graphics, vol 23(3), pp 441–448, August 2004
19. Foster N, Metaxas D (1996) Realistic animation of liquids. Graph Models Image Process 58(5):471–483
20. Fedkiw R, Stam J, Jensen HW (2001) visual simulation of smoke. In: Proceedings of SIGGRAPH, 15–22
21. Goktekin TG, Bargteil AW, O'brien JF (2004) A method for animating viscoelastic fluids. In: Proceedings of ACM SIGGRAPH 2004. ACM transactions on graphics, vol 23(3), pp 463–468, August 2004
22. Greenwood S, House D (2004) Better with bubbles: enhancing the visual realism of simulated fluid. In: ACM SIGGRAPH/Eurographics symposium on computer animation, pp 287–296
23. Gueyffier D, Li J, Nadim A, Scardoveli R, Zaleski S (1999) Volume-of-fluid interface tracking with smoothed surface stress method for three-dimensional flows. J Comput Phys 152(2):423–456
24. Hu X, Adams N (2006) A multi-phase SPH method for macroscopic and mesoscopic flows. J Comput Phys 213(2):844–861
25. Hong J-M, Kim C-H (2003) Animation of bubbles in liquid. In: Computer Graphics Forum (Eurographics 2003 Proceedings), vol 22(3), pp 253–262, Sept 2003
26. Hong J-M, Kim C-H (2004) Controlling fluid animation with geometric potential. Comput Animat Virtual Worlds 15(3–4):147–157
27. Hong J-M, Kim C-H (2005) Discontinuous fluids. In: Proceedings of ACM SIGGRAPH 2005. ACM transactions on graphics (TOG), vol 24(3), pp 915–920, July 2005
28. Hong J-M, Lee H-Y, Yoon J-C, Kim C-H (2008) Bubbles alive. In: Proceedings of ACM SIGGRAPH 2008. ACM transactions on graphics (TOG), vol 27(3), pp 48:1–48:4, Aug 2008
29. Hirt CW, Nichols BD (1981) Volume of fluid (VOF) method for the dynamics of free boundaries. J Comput Phys 39(1):201–255
30. Homsy GM (1987) Viscous fingering in porous media. Annu Rev Fluid Mech 19:271–311
31. Hong J-M, Shinar T, Fedkiw R (2007) Wrinkled flames and cellular patterns. In: Proceedings of ACM SIGGRAPH 2007. ACM transactions on graphics (TOG), vol 26(3), pp 471–476, July 2007
32. Harlow FH, Welch JE (1965) Numerical calculation of time-dependent viscous incompressible flow of fluid with free surfaces. Phys Fluids 8:2182–2189
33. Ianniello S, Mascio AD (2010) A self-adaptive oriented particles Level-Set method for tracking interfaces. J Comput Phys 229(4):1353–1380
34. Jo E-C, Kim D-Y, Song O-Y (2011) A new SPH fluid simulation method using ellipsoidal kernels. J Vis 14(4):371–379
35. Kim J, Cha D, Chang B, Koo B, Ihm I (2006) Practical animation of turbulent splashing water. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 335–344
36. Kang M, Fedkiw RP, Liu X-D (2000) A boundary condition capturing method for multiphase incompressible flow. J Sci Comput 15(3):323–360
37. Kass M, Miller G (1990) Rapid, stable fluid dynamics for computer graphics. In: ComputerGraphics (Proceedings of ACMSIGGRAPH'90), vol 24, pp 49–57
38. Kim P-R, Lee H-Y, Kim J-H, Kim C-H (2012) Controlling shapes of air bubbles in a multi-phase fluid simulation. Vis Comput 28(6–8):597–602
39. Kim B, Liu Y, Llamas I, Jiao X, Rossignac J (2007) Simulation of bubbles in foam with the volume control method. In: Proceedings of ACM SIGGRAPH 2007. ACM transactions on graphics (TOG), vol 26(3), pp 98:1–98:9, July 2007
40. Kim B, Liu Y, Llamas I, Rossignac J, Flowfixer (2005) Using BFECC for fluid simulation. In: Proceedings of Eurographics workshop on natural phenomena, pp 51–56

41. Kim B, Liu Y, Llamas I, Rossignac J (2007) Advections with significantly reduced dissipation and diffusion. IEEE Trans Vis Comput Graph 13(1):135–144
42. Kuck H, Vogelgsang C, Greiner G (2002) Simulation and rendering of liquid foams. Graph Interface
43. Kunimatsu A, Watanabe Y, Fujii H, Saito T, Hiwada K, Takahashi T, Ueki H (2001) Fast simulation and rendering techniques for fluid objects. Computer Graphics Forum (Proceedings of Eurographics 2001) 20(3):357–367
44. Lorensen WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of ACM SIGGRAPH'87, vol 21(4), pp 163–169, July 1987
45. Liu X-D, Fedkiw RP, Kang M-J (2000) A boundary condition capturing method for poisson's equation on irregular domain. J Comput Phys 172(1):71–98
46. Losasso F, Fedkiw R, Osher S (2005) Spatially adaptive techniques for level set methods and incompressible flow. Comput Fluids
47. Losasso F, Gibou F, Fedkiw R (2004) Simulating water and smoke with an octree data structure. In: Proceedings of ACM SIGGRAPH 2004. ACM Transactions on Graphics (TOG), vol 23(3), pp 457–462, Aug 2004
48. Lee H-Y, Hong J-M, Kim c-h (2005) Interchangeable SPH and level set method in multiphase fluids. Vis Comput 25:713–718
49. Losasso F, Irving G, Guendelman E, Fedkiw R (2006) Melting and burning solids into liquids and gases. IEEE Trans Vis Comput Graph 12(3):343–352
50. Liu MB, Liu g r, Lam KY (2006) Adaptive smoothed particle hydrodynamics for high strain hydrodynamics with material strength. Shock Waves 15(1):21–29
51. Losasso f, Shinar t, Selle A, Fedkiw R (2006) Multiple interacting liquids. In: Proceedings of ACM SIGGRAPH 2006. ACM transactions on graphics (TOG), vol 25(3), pp 812–819, July 2006
52. Losasso f, Talton j, Kwatra n, Fedkiw R (2008) Two-way coupled SPH and particle level set fluid simulation. IEEE Trans Vis Comput Graph 14(4):797–804
53. Magnaudet J, Eames I (2000) The motion of high reynolds-number bubbles in inhomogeneous flows. Annu Rev Fluid Mech 32:659–708
54. Mcnamara A, Treuille A, Popovic Z, Stam J (2004) Fluid control using the adjoint method. In: Proceedings of ACM SIGGRAPH 2004. ACM transactions on graphics (TOG), vol 23(3), pp 449–456, Aug 2004
55. Mendelson HD (1967) The prediction of bubble terminal velocities from wave theory. Adv Chem Eng J 13(2):250–253
56. Mihalef V, Metaxas D, Sussman M (2007) Textured liquids based on the marker level set. Comput Graph Forum 26(3):457–466
57. Mihalef V, Unlusu B, Metaxas D, Sussman M, Hussaini MY (2006) Physics based boiling simulation. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation 1:317–324
58. Mullen P, Mckenzie A, Tong Y, Desbrun M,A (2007) Variational approach to eulerian geometry processing. In: Proceedings of ACM SIGGRAPH 2007. ACM transactions on graphics (TOG), vol 26(3), pp 66:1–66:10, July 2007
59. Müller M, Charypar D, Gross M (2003) Particle based fluid simulation for interactive applications. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation pp 154–159
60. Müller M, Solenthaler B, Keiser R, Gross M (2005) Particle-based fluid-fluid interaction. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 237–244
61. Nguyen D, Fedkiw R, Jensen H (2002) Physically based modeling and animation of fire. In: Proceedings of ACM SIGGRAPH 200. ACM transactions on graphics (TOG), vol 21(3), pp 721–728, July 2002
62. O'Brien J, Hodgins J (1995) Dynamic simulation of splashing fluids. Proc Comput Animat 95:198–205

63. Osher S, Fedkiw R (2002) Level set methods and dynamic implicit surfaces. Springer, New York
64. Park J, Kim Y, Wi D, Kang N, Shin SY, Noh J (2008) A unified handling of immiscible and miscible fluids. Comput Animat Virtual Worlds 19(3–4):455–467
65. Premoze S, Tasdizen T, Bigler J, Lefohn A, Whitaker R (2003) Particle-based simulation of fluids. In: Computer Graphics Forum (Eurographics Proceedings), vol 22, pp 401–410
66. Rasmussen N, Enright D, Nguyen D, Marino S, Sumner N, Geiger W, Hoon S, Fedkiw R (2004) Directable photorealistic liquids. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 193–202
67. Rider WJ, Kothe DB (April 1998) Reconstructing volume tracking. J Comput Phys 141(2):112–152
68. Rahman A, Manzur M (2008) Dynamic texture synthesis using motion distribution statistics. J Res Pract Inf Technol 40(2):129–148
69. SAAD Y (1996) Iterative methods for sparse linear systems. PWS Publishing
70. Schechter H, Bridson R (2008) Evolving sub-grid turbulence for smoke animation. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 1–7
71. Son G, Dhir VK, Ramanujapu N (1999) Dynamics and heat transfer associated with a single bubble during nucleate boiling on a horizontal surface. J Heat Transf 121(3):623–631
72. Shin S, Juric D (2002) Three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. J Comput Phys 180(2):427–470
73. Shin S-H, KIM C-H (2007) Target-driven liquid animation with interfacial discontinuities. Comput Animat Virtual Worlds 18(4–5):447–453
74. Shew WL, Pinton J-F (2006) Dynamical model of bubble path instability. Phys Rev Lett 97:144508
75. Selle A, Rasmussen N, Fedkiw R (2005) A vortex particle method for smoke, water and explosions. In: Proceedings of ACM SIGGRAPH 2005. ACM transactions on graphics (TOG), vol 24(3), pp 910–914, July 2005
76. Song O-Y, Shin H-C, Ko H-S (2005) Stable but non-dissipative water. ACM Trans Graph 24(1):81–97
77. Stam J (1999) Stable fluids. In: Proceedings of ACM SIGGRAPH, pp 121–128
78. Sussman M (2003) A second order coupled level set and volume of-fluid method for computing growth and collapse of vapor bubbles. J Comput Phys 187(1):110–136
79. Torres DJ, Brackbill JU (2000) The point-set method: front-tracking without connectivity. J Comput Phys 165(2):620–644
80. Tryggvason G, Bunner B, Esmaeeli A, Juric D, Al-Rawashi N, Tauber W, Han J, Nas S, Jan Y-J (2001) A front tracking method for the computations of multiphase flow. J Comput Phys 169(2):708–759
81. Takahashi T, Fujii H, Kunimatsu A, Hiwada K, Saito T, Tanaka K, Ueki H (2003) Realistic animation of fluid with splash and foam. Computer Graphics Forum (In: Proceedings of Eurographics 2003), vol 22(3), pp 391–400, Sept 2003
82. Thüerey N (2007) Physically based animation of free surface flows with the lattice Boltzmann method. Ph.D. thesis, University of Erlangen-Nuremberg
83. Treuille A, Mcnamara A, Popović Z, Stam J (2003) Keyframe control of smoke simulations. In: Proceedings of ACM SIGGRAPH 2003. ACM transactions on graphics (TOG), vol 22(3), pp 716–723, July 2003
84. Thüerey N, Sadlo F, Schirm S, Müller-Fischer M, Gross, M (2007) Real-time simulations of bubbles and foam within a shallow water framework. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, pp 191–198
85. Unverdi S, Tryggvason G (1992) A front tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 100(1):25–37
86. Wojtan C, Carlson M, Muycha PJ, Turk G (2007) Animating corrosion and erosion. Eurographics workshop on natural phenomena, pp 15–22
87. Wojtan C, Thürey N, Gross M, Turk G (2009) Deforming meshes that split and merge. ACM Trans Graph 28(3)

88. Wojtan C, Thürey N, Gross M, Turk G (2010) Physics-inspired topology changes for thin fluid features. In: Proceedings of ACM SIGGRAPH 2010. ACM transactions on graphics (TOG), vol 29(4), July 2010
89. Yu Y, Ho Jung, Cho H (1999) A new water droplet model using metaball in the gravitational field. Comput Graph 23(2):213–222
90. Yu J-H, Turk G (2010) Reconstructing surfaces of particle-based fluids using anisotropic kernels. In: Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, p 217
91. Zhu H, Liu X, Liu Y, Wu E (2006) Simulation of miscible binary mixtures based on lattice Boltzmann method: research articles. Comput Animat Virtual Worlds 17(3–4):403–410
92. Zheng Z, Yang J, Zhu Y (2006) Face detection and recognition using colour sequential images. J Res Pract Inf Technol 38(2)
93. http://developer.nvidia.com/

http://www.springer.com/978-981-287-486-3

Real-Time Visual Effects for Game Programming

Kim, C.-H.; Kim, S.-J.; Kim, S.-K.; Kang, S.-J.

2015, XII, 227 p. 166 illus., 85 illus. in color., Hardcover
ISBN: 978-981-287-486-3