Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Better with Bubbles: Enhancing the Visual Realism of Simulated Fluid

S. T. Greenwood and D. H. House

Visualization Laboratory, Texas A&M University, USA

Abstract

We present a method for including the visual effect of bubbles in a computer graphics fluid simulation, thus enhancing the illusion of realism for a splashing fluid. Previous fluid simulation methods have not included bubbles. Bubble creation is integrated into the particle level-set fluid simulation algorithm. Individual bubbles are approximated by spheres, which form more complex shapes where they intersect. The rendering of bubbles and fluid are integrated to create the appearance of one continuous surface. At the fluid-air boundary, we integrate bubbles whenever level-set marker particles pass from from the outside to the inside of the fluid. Thus, these particles represent air that has become trapped within the fluid surface. In addition, we detect empty pockets within the fluid, that are often formed due to turbulence, and create bubbles within this space. This is an inexpensive way of giving the impression that the air trapped in air pockets has become bubbles. Photo-realistic images of simulation results are rendered with a raytracer that has been enhanced to include caustics, and to handle bubble-bubble interfaces. Comparison of these images with images rendered without bubbles supports our position that the simple addition of bubbles to a fluid simulation greatly enhances visual realism.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Raytracing, I.6.8 [Simulation and Modeling]: Visual

1. Introduction

Since we experience water in so many ways in our lives, simulating water for computer graphics is both important and highly challenging. Recent methods have produced successively more realistic results but still look somewhat artificial. Figure 1 illustrates our thesis that the visual appeal of splashing water can be greatly enhanced by the inclusion of bubbles.

Our approach is to integrate the automatic production of bubbles into a fluid simulation, and then to move these bubbles with the water volume or its surface. We exploit marker particles, that are used to maintain detail in the fluid surface level set. Some of these particles escape out of the surface and in previous work have been used to generate splash droplets. We do the converse. Particles escaping from the air into the fluid represent air volume that has been trapped inside the liquid, indicating where bubbles might form. In addition, we detect trapped air pockets in the fluid, collapse them, and convert them into bubbles.

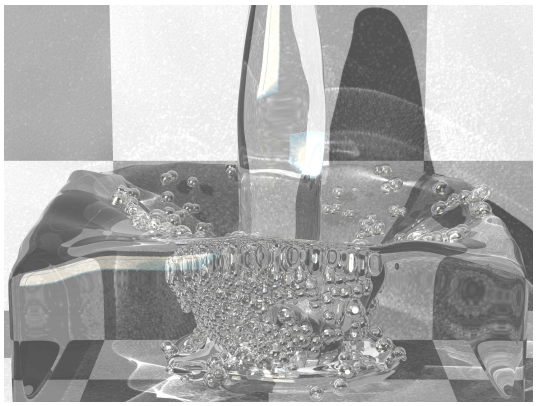


Figure 1: *Splashing water with bubbles.*

2. Previous Work

There has been considerable earlier work on fluid simulation and some on bubbles and foam. The method presented

in this paper synthesizes approaches in these two directions, providing for the automatic integration of bubbles and foam into a simulated fluid. The work intentionally avoids simulating bubble formation using a surface tension model, in favor of a more computationally tractable approach that is only first-order physically reasonable, while producing visually appealing results. Thus the extensive literature (for example see [BKZ92, LNS*94]) related to bubbles and surface tension is not surveyed.

2.1. Fluid simulation

The use of the Navier-Stokes equations for the simulation of complex water motion was introduced to the computer graphics community by Foster and Metaxes [FM96]. They used the marker and cell method of Harlow and Welch [HW65] to create 3D animations of water. Marker particles were distributed throughout the computational grid of the simulation and then transported by the velocity field of the fluid. Cells with at least one particle were treated as fluid cells, with surface cells being the set of all fluid cells adjacent to at least one empty cell. Chen et al. [CJR95] improved this method by placing marker particles only near the surface of the fluid. Stam [Sta99] introduced a semi-lagrangian treatment of the advection portion of the Navier-Stokes equations allowing large timesteps without causing instability. Foster and Fedkiw [FF01] introduced a hybrid model that combined a level set with surface marker particles. The level set is defined implicitly by a function ϕ of spatial position, whose magnitude measures distance from the surface and whose sign is positive outside of the fluid and negative inside. The level set maintained the overall shape of the surface, while the particles were used to maintain surface detail. The markers and level set were advanced in smaller timesteps than that used for the fluid in order to reduce the error in the representation of the surface. Most recently, Enright et al. [EMF02] proposed the use of additional marker particles *outside* of the liquid volume in order to maintain surface detail coming from columns of air formed in the fluid. This is the method that we use for our fluid simulations.

2.2. Bubbles and foam

Bubbles and foam have been studied extensively by mathematicians, mainly because of their surface minimizing properties. Analytical solutions for the geometry of bubble clusters up to the size of three bubbles have been found. Glassner [Gla00] modeled these groups with CSG operations. Numerical methods must be used for larger bubble clusters, since their liquid films are not spherical. Bolton simulated foams in two dimensions as a network of curved films [Bol90]. Hong and Kim [HK03] simulated the behavior of individual bubbles for computer graphics applications by integrating bouyancy forces to correct the fluid velocity field. However, they did not treat the interaction of bubbles or the creation of foams. Durian created a model that uses a group of interacting bubbles to represent foam [Dur95, Dur97]. Differing from the network based approaches, Durian's method simplifies the simulation of the foam because it does not have to deal with changes in topology. Using similar techniques, Kück et al. simulated and rendered foams in 3 dimensions [KVG02]. Our method for the simulation and rendering of bubbles is largely based on their techniques.

3. Creating Bubbles from Escaped Marker Particles

In the real world, bubbles are created whenever air is trapped inside of a fluid. When these bubbles reach the surface, they persist because of the surface tension in the film of water surrounding the bubble acting against the pressure within the bubble. It is possible to add surface tension effects to fluid simulations. The air could be simulated as a second fluid, and bubbles could be simulated along with the level set and underlying velocities. However, the film of the bubble when it reaches the surface is too fine to be simulated with the underlying computational gird typically used for fluid simulation.

In our method, we avoid the need for the fine detail necessary to simulate bubbles by simply creating spherical bubble objects that are moved by the velocities within the fluid simulation. Our bubbles are passive, and have no effect on the underlying fluid simulation. Our assumption is that in a moving liquid, the effects of relatively small bubbles are not significant enough to cause a noticeable change in the behavior of the fluid. At the surface, a single bubble has miniscule mass compared to the water itself, so it will not significantly disturb water that is moving. If the water is still, a single bubble could cause noticeable ripples, but this is not typically a problem that we would address with a full Navier Stokes simulation.

Since a bubble is part of the fluid, when it moves the motion one sees simply reflects the motion of the surface of the fluid. Since we represent this detail of the surface with the bubble object, the bubble object moves instead of the level set representation of the fluid (which is too coarse to represent a thin bubble film). Using passive bubbles is simpler and is more practical than the alternative of allowing the bubbles to create forces that directly affect fluid velocities. Once the fluid simulation is run and saved, the bubble simulation can be tweaked to the needs of the animator. If the bubbles did affect the motion of the fluid, any change to the bubble configuration would require rerunning the fluid simulation.

3.1. Bubble creation

Our method of bubble creation is integrated into a fluid simulation based on the method of Enright et al. [EMF02]. We create bubbles whenever air marker particles cross the fluid-air interface and get trapped inside the fluid. When a particle that represents air moves too far across the interface, it is a

good indication of where there might be mixing of air and water. The bubble simulation created is independent of how the Navier-Stokes equations are solved, but it depends on the hybrid particle-level set method for the representation of a fluid surface, which uses marker particles representing the outside of the fluid.

Since the air is not simulated in the particle-level set method, air pockets that form are ignored by the fluid simulation and are simply engulfed by fluid. If this is noticeable, it is very undesirable. In reality, the air pockets would become bubbles and would not lose volume. If we did not detect air pockets, the air pocket would shrink, as the fluid velocities flow inward. This would push the marker particles into tighter and tighter spaces, and when the air pocket is finally totally empty, a few bubbles would form from the escaped marker particles. That much volume loss is very unrealistic, and created a challenge for our method.

Fortunately, it is straightforward to detect these air pockets by looking for cells that do not contain fluid and are not connected to the air outside of the fluid. This is achieved by a flood fill algorithm [FvDFH90]. The cells are treated like pixels. Those with no fluid are "painted" one "color," which designates them as air cells. All other cells, including wall cells, are "painted" another "color". The fill algorithm is started from an empty cell that is guaranteed to be in the atmosphere (such as a cell at the top of the simulation). Thus, all "painted" cells are considered part of the atmosphere. Empty cells that are not "painted" the atmosphere "color" are considered parts of air pockets. Figure 2 illustrates the only modification needed to the basic seed fill algorithm. This is that cells diagonally adjacent to an atmosphere cell should be included only if the level set implicit function value ϕ between the cells is positive (i.e. outside of the surface). If ϕ is positive, the air extends diagonally between cells, as seen in the left hand diagram. Otherwise, the diagonal of the cell is blocked off with fluid, as seen in the right hand diagram.

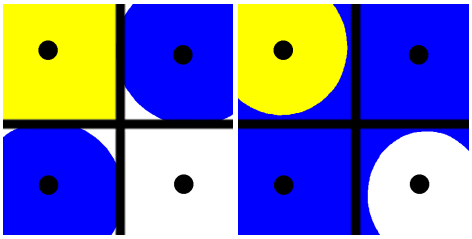


Figure 2: *Diagonal Inclusion Cases*

Atmosphere cells are marked in yellow. The fluid is marked in blue. Case when diagonal cell should be included with atmosphere (left) and when it should not (right).

During simulation, the air pockets are detected before the level set is reinitialized and after it is moved and corrected. When an air pocket is detected, we convert the empty cells to fluid by making the level set implicit function value ϕ negative (i.e. inside the fluid). Since the marker particles within these cells are now far from the interface, they are considered escaped and become bubbles. This gives the visual appearance that the air pocket changes into bubbles. The velocities of the cells in the air pockets are initialized to reasonable values through the extrapolation of surface velocities in the surface conservation step of the particle-level set method.

3.2. Avoiding unrealistic bubbles

We found that we needed to avoid creating bubbles from escaped marker particles in cases where it would be unlikely for bubbles to form. One precaution taken is to create a bubble only if the curvature of the surface is negative at the position of the particle. Since bubbles are supposed to represent trapped air, they should not be formed when there is a positive curvature (i.e convex outward).

In the particle level set method, particles are assigned radii, and these radii are adjusted in order to keep the surface smooth. It is more probable that particles of the minimum radius will be far enough from the level set so that they are considered "escaped." Our experiments showed that, in many of these situations, it is not appropriate for bubbles to form, so we use caution when using these small particles to create bubbles. Since a particle with radius larger than the minimum represents a larger move of the surface, we always accept these particles. Bubbles that are created from particles with the minimum radius are marked. Once all of the bubbles are created, we check that new bubbles created from small particles are in contact with those created from larger particles. If there is no contact, then we remove the small-particle bubbles.

3.3. Bubble size

In our method, the placement and size of bubbles is initially based on the position and radius of escaped marker particles. In the particle-level set method, the radius of a particle is always between 0.1 and 0.5 of the cell width. Thus, chosing bubble radius directly from particle radius would create extremely tiny bubbles for fine grids. Additionally, if the escaped particle's radius were simply scaled up by a constant value, there would not be enough variation in bubble size.

Bubble radii could simply be generated randomly, but there is valid information in the marker particle radii. For common splashing and sloshing, a good proportion of the escaped marker particles are small. For bubbles created from air pockets, a significant proportion of the escaped marker particles have a radius of the maximum size. It makes sense for large air pockets to create larger bubbles, so bubbles created from marker particles with the maximum radius should have a larger radius on average than bubbles created from marker particles with the minimum radius.

Our method generates bubble radii based on a gaussian

distribution with mean and standard deviation determined by the radius of the marker particle. Particles with the maximum radius create bubbles with one mean and standard deviation. Particles with the minimum radius create bubbles with smaller mean and standard deviation. We interpolate between these extremes based on particle radius. Random values that are too large or small are recalculated until they are an appropriate size (as determined by user defined parameters). It is important that bubbles with sizes that are too small to be sampled effectively by normal raytracing should be avoided (unless there is a special case for rendering these bubbles). This method effectively creates bubbles of varied sizes related to the radius of the marker particle. As shown in Figure 3, this method is independent of the cell width and grid size, so that similar bubbles will be created for different grid resolutions.

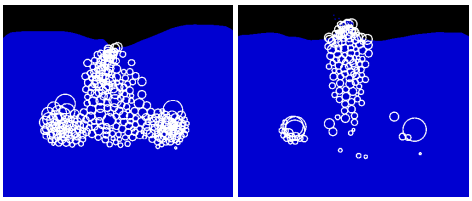


Figure 3: *Bubbles created in different grid resolutions*

30x30 grid (left) and 60x60 grid (right), note that bubble sizes are similar.

3.4. Bubble merging and popping

It should be noted that the radii of adjacent marker particles can overlap. Further, if the bubbles are larger than the original marker particles, then there can be significant overlap in the created bubbles. During bubble creation, it might make sense to merge heavily overlapping bubbles into larger bubbles. Also, bubbles that meet in the interior of the fluid might merge into larger bubbles. We chose to ignore these situations. Since we model bubbles with spheres, larger bubbles are visually undesirable (as will be discussed in the last section). We do remove bubbles that are completely encompassed by other bubbles.

A real bubble pops when one of its surface films drains too much, becomes thin and breaks. This can be modeled by removing bubbles randomly. The larger the surface area, the more likely it is that a bubble is going to pop, so larger bubbles pop sooner than smaller bubbles. When a bubble is adjacent to an obstacle, there is less surface area of film and thus less area to break. We let the lifetime of a bubble be

$$L = \begin{cases} 0, & r_b \leq r_s \\ \alpha \frac{(r_b - r_s)}{(r_l - r_s)}, & r_b > r_s \end{cases}, \qquad (1)$$

where r_b is the bubble's radius, r_s is the minimum radius, r_l the maximum radius, and α is a tunable scale factor. Then at each timestep in the simulation, the probabality that a bubble is removed is

$$\min(\frac{(1+L)h}{T}, 1.0),$$

where T is the bubble's current lifetime and h the simulation timestep. Bubbles in contact with obstacles tend to last longer, so we use a different lifetime parameter α for bubbles in contact with walls. Since it is not possible for bubbles to "pop" while inside the fluid, only bubbles that have reached the surface are candidates for removal.

3.5. Particle density and creation of bubbles

Since there are different densities of marker particles per volume, we decided to make the number of bubbles formed dependent on volume and not on grid resolution. A similar fluid simulation with twice the resolution in all dimensions has eight times as many maker particles per unit volume for the same number of marker particles per cell. This is a problem as the finer grid would tend to create more bubbles than the coarser grid. For rapid testing, it is especially useful to work with coarser grid sizes, so consistency across resolutions is important. A marker/volume density is chosen in which all escaped particles are considered to create bubbles. In the step where bubbles are created, the ratio between the desired and the actual marker/volume densities determines the probability of bubble creation. This assures that for a simulation with eight times the desired marker particle density, we only use one on average eighth of the escaped marker particles to create bubbles.

4. Bubble Simulation

4.1. Simulation of foams

We based our foam simulation on the methods outlined by Kück et al. [KVG02]. This subsection outlines their approach. The exact geometry of foam films is not simulated. Instead, bubbles are simulated by spheres of fixed radii and are moved according to a set of simulated forces.

As shown in Figure 4, when bubbles are in contact, attractive and repelling spring forces are created to cause them to overlap and appear to be part of a foam structure. The attractive and repelling forces are set so that they cancel each other out when the bubbles overlap by the desired amount. Similar attractive and repelling forces are applied between bubbles and obstacles.

When two bubbles are in contact, the repelling force of bubble i on bubble j is

$$F_{ij}^r = k_r(\frac{1}{||p_i - p_j||} - \frac{1}{l_{ij}})(p_i - p_j), \qquad (2)$$

where k_r is a user defined strength coefficient, p_i and p_j are the positions of bubbles i and j, and $l_{ij} = r_i + r_j$ is the rest distance between the bubbles.

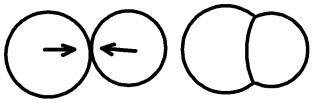


Figure 4: *Attractive forces acting on touching bubbles.*

The attractive force

$$F_{ij}^a = -k_a c_n c_d (p_i - p_j)/||p_i - p_j||, \quad (3)$$

between bubble i and bubble j depends on how many other bubbles a bubble is in contact with. If N_i is the set of bubbles overlapping sphere i, the coefficient

$$c_n = \frac{1/|N_i| + 1/|N_j|}{2}.$$

This makes the attraction force smaller for bubbles in large clusters. Also, the coefficient

$$c_d = \frac{||p_i - p_j|| - \max(r_i, r_j)}{\min(r_i, r_j)}.$$

makes the attractive force a nonlinear function of distance. It is zero when the center of a smaller bubble rests on the edge of a larger bubble, and increases when the distance is larger. It becomes negative when the larger bubble encompasses the center of the smaller bubble, changing the attraction force to a repelling force that prevents bubbles from overlapping too much.

The viscous force on bubble i from other bubbles is

$$F_i^v = k_v(\bar{v_i} - v_i), \quad (4)$$

where k_v is an adjustable parameter, v_i is the velocity of bubble i and $\bar{v_i}$ is the average velocity of bubbles in contact with bubble i. The friction force between bubble and a barrier object is defined similarly. Gravity is a constant force F_g acting on all bubbles. Air damping is

$$F_i^d = -k_d v_i, \quad (5)$$

where k_d is a user-defined parameter.

It is assumed that the bubbles are massless, which means that the sum of all forces must be zero, and allows a direct solution for velocity. From Equations 2 through 5 we get the explicit solution

$$v_i = \frac{k_v \bar{v_i} + k_o \bar{v_i^o} + F_s + F_g}{k_v}, \quad (6)$$

where F_s is the sum of all attracting and repelling "spring" forces.

4.2. Coupling bubble and fluid simulations

In order to incorporate the bubble simulation model of Kück et al. [KVG02] into the fluid simulation, we have to add forces from the fluid on the bubbles.

Bubbles within the fluid are subject to a force due to fluid pressure. Foster and Metaxes [FM96] modeled buoyant objects that were subject to a force related to the negative gradient of the pressure times the volume. In our model the pressure force on bubble i is

$$F_i^p = -K_p \nabla p_i V_i, \quad (7)$$

where p_i is the pressure at the position of the bubble, V_i is the bubble's volume, and K_p is a user defined parameter that adjusts the contribution of pressure on the bubble's motion. Since our forces are modeled linearly, for large bubbles, velocities can get unrealistically large. For the calculation we clip the volume V_i to a user defined limit to keep bubbles from moving unrealistically fast.

The other fluid force on bubbles is due to the viscosity of the fluid. The viscous force on bubble i is

$$F_i^v = K_v(u_i - v_i), \quad (8)$$

where u_i is the velocity of the fluid at the position of the bubble, and K_v is a user-defined coefficient of viscosity.

4.3. Bubble forces in different areas

Depending on a bubble's location with respect to the fluid's surface, different forces will be acting. Figure 5 shows the four distinct regions that we consider: below, adjacent below, adjacent above, and above the surface.

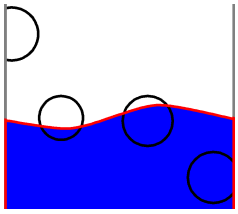


Figure 5: *Bubble regions.*

Above the surface (left), adjacent above (middle left), adjacent below (middle right), and below the surface (right). Fluid surface interface is marked in red.

A bubble is considered to be in the *below surface* region when the level set implict function value ϕ at the position of the bubble is below a negative multiplier of the bubble's radius. This means that the bubble must be a smaller distance than its own radius from the surface to be considered at the surface. The multiplier must be between 0 and -1. We used -0.1 which seemed to work well. If the bubble's negative radius were used as the threshold value of ϕ, a bubble below the surface that is adjacent to an obstacle would incorrectly

be considered to be at the surface, as in the rightmost bubble in Figure 5. Below the surface of the fluid there is only a repulsion force between bubbles [KVG02]. In addition, there is no friction force between bubbles or objects, and gravity will not have significant impact. Thus, below the surface, the solution for velocity is

$$v_i = \frac{1}{K_v}(K_v u_i + F_i^s + F_i^p), \tag{9}$$

where F^s omits all attractive forces.

If the value of ϕ at a bubble's position is below zero but greater than the negative of the bubble's radius, then that bubble is in the *adjacent below the surface* region. When bubbles are at the surface, there is an attraction force between bubbles that are in contact, and their friction is no longer negligible. Viscous interactions with other bubbles cause a bubble's velocity to be dependent on the previous velocities of its neighboring bubbles. This dependency on previous states is undesirable with the massless assumption. Instead, we use the current velocities of the fluid at the position of the adjacent bubbles instead of the velocity of the adjacent bubbles from the previous timestep. Thus, the friction force is

$$F_i^f = K_f(\bar{u_i} - v_i) \tag{10}$$

where $\bar{u_i}$ is the average of the fluid velocities at bubbles intersecting bubble i. Averaging the fluid velocities rather than the bubble velocities is reasonable at the surface, since there a bubble's velocity is most influenced by viscous interaction with the fluid at that point. Here, pressure simply pushes the bubble up to the surface. Thus, bubble velocity adjacent below the surface is

$$v_i = \frac{K_v u_i + K_f \bar{u_i} + F_i^s + F_i^p}{K_v + K_f}. \tag{11}$$

If the bubble's position is between zero and the bubble's radius, then that bubble is in the *adjacent above the surface* region. The calculation of velocity is the same as in the adjacent below region except that F^g (gravity) is substituted for F^p (pressure) in Equation 11.

Once the value of ϕ at a bubble position is greater than that bubble's radius, the bubble is no longer in contact with the fluid. These bubbles may be "popped" because water bubbles cannot persist away from the water's surface.

4.4. Simulation scheme

The particle-level set surface is updated on a sub-cycle of the timestep used to advance the Navier-Stokes equations, and the bubble simulation is a sub-cycle of the timestep used to update the fluid surface. This means that the timestep used while simulating the bubbles may be smaller than the timestep used to move forward the level set and marker particles. The timestep h that we use is regulated by the CFL condition

$$h < r_{\min}/v_{\max}, \tag{12}$$

where $r_{\min}$ is the smallest bubble radius, and $v_{\max}$ is the largest bubble velocity. For simulating our bubbles we used an Euler timestep 100 times smaller than this CFL condition. Since the simulation of the bubbles is much faster than the fluid simulation, it did not significantly increase run times, and this guarantees that bubbles cannot miss contact with each other. The bubble simulation step comes after the reinitialization of the level set. At this point, all of the new bubbles for this timestep have been created. When the level set is advanced, we save the old level set. This allows us to know the current value of the level set implicit function ϕ at any point in the bubble subcycle by interpolating between previous and past values. The more accurately we know where the surface of the fluid is during a timestep the better, as the forces applied to a bubble vary drastically in relation to its position relative to the fluid surface. This is because the transition of a bubble between different regions is treated as a discrete event.

5. Bubble Rendering

5.1. Rendering in Kück et al.

Kück et al. [KVG02] render a contiguous foam structure from an arbitrary configuration of spherical bubbles as shown in Figure 6. This subsection explains their method. The foam is ray traced and calculations are made at each intersection of a ray with the spheres. With time saving approximations, they succeeded in rendering foams on the order of several thousand bubbles, as seen from a medium distance, in reasonable time.

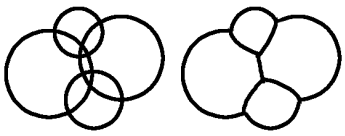


Figure 6: *Making spheres appear to be foam.*

Sphere representation (left) and corresponding foam structure (right).

No refraction is calculated at a bubble film. The assumption is that the film is too thin to noticeably change the direction of a light ray. They use an approximation for fresnel reflections to shade the bubble's film. The Fresnel term is the ratio of reflected to refracted non-polarized light from a dielectric (non-conducting surface) [FvDFH90]. This approximation to the fresnel term is

$$f = 1 - N \cdot I, \tag{13}$$

where N and I are unit vectors in the direction of the surface normal and the incident light. Figure 7 demonstrates that this

creates a smooth transition from totally reflective to totally refractive behavior.

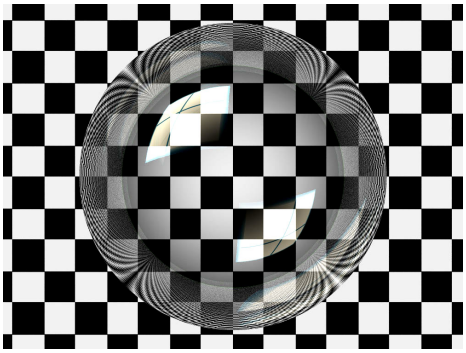


Figure 7: *Bubble shader using approximated Fresnel term.*

When two or more bubbles overlap, their goal is to create the appearance of a contiguous foam structure. When a ray passes through two overlapping spheres, the film between the them is approximated. The approximated film's position is taken to be the average of the two hit points on the spheres, and the normal is the average of the normals at the hit points.

No shading calculations are done at the actual surfaces of the spheres where they overlap. This appears correct because with the higher curvature of the smaller bubble, the averaged normals give the impression of the surface curved toward the larger bubble. This approximation is inexpensive and works for most viewing angles. As can be seen in Figure 8, these methods are effective at creating the appearance of two bubbles joined together.

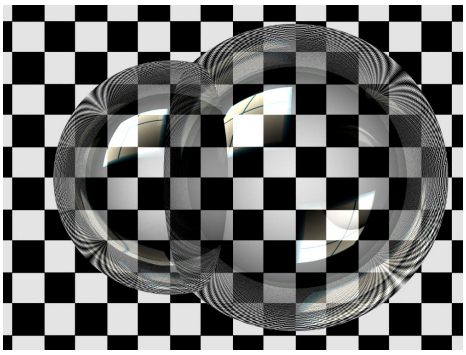


Figure 8: *Two-bubble cluster.*

Where three bubbles overlap, Kück et al. define a plateau border, and render this region using an ambient term to represent the light that would be heavily scattered in this region, as well as a term that represents refracted light on the plateau border. In this method, the ray stops once it reaches the plateau border. This works since they were interested in dense foams that heavily scatter light and were not interested in smaller bubble clusters.

5.2. Rendering bubbles with fluid

Our simulation of bubbles in fluid differs from the simulation of foam because foams consist of thousands of bubbles. We deal with only hundreds, and they are viewed at closer range (i.e. they are bigger) than in foam simulations. Also, our bubbles are sloshed around vigorously.

For the integration of approximated films between two bubbles and transparent objects, a special case needs to be addressed. For the case shown in Figure 9, the approximated film is partially inside of an object. We detect this case and do not render it.

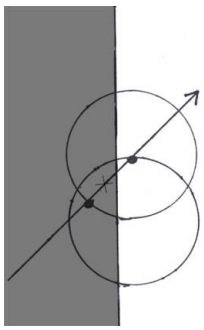


Figure 9: *Two-bubble case inside of object.*

For two overlaping bubbles, it must be checked whether the approximated film is behind the surface of an object.

For our purposes, dealing with small bubble clusters, the use of an ambient term when three bubbles overlap is undesirable because it results in a noticable visual artifact in close or mid-distance views. We wanted to use the approximated films approach that we used with two bubbles so that they met somewhere in the three bubble overlap region. Unfortunately, since the films between two bubbles are not explicitly defined, there is no easy way to do this.

Instead we treated the three bubble overlapping case similarly to the two bubble overlapping case. As seen in Figure 10b, once a ray enters an area where the bubbles overlap, no shading is calculated until the ray exits into a single bubble. Then the normal is set to the average between this hit point, and the original hit point when the ray entered the second bubble. The reflection ray leaves the same point as the refraction ray, because there is no simple way of knowing which bubbles the average of the hit points is inside. (In the two bubble case, the average point is inside of the two bubbles so we can send the reflection ray from that point).

The original hit point was used for approximating further film intersections around the three bubble region (as the red reflection ray may create in Figure 10). While this method is simply a heuristic, it allows the ray to continue past the three-bubble region and does not draw attention to itself. Figure 11 shows the visual quality obtained when using this approximation.

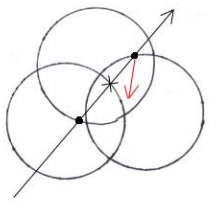


Figure 10: *Modified method for shading where three bubbles overlap.*

Reflection ray (red) is sent from the final hit point because it is unknown which bubbles the averaged point (x) is inside.

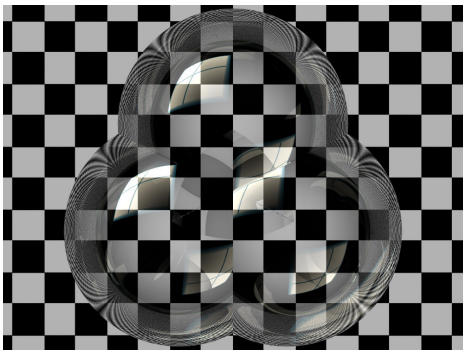


Figure 11: *Three-bubble cluster.*

The amount of light that is reflected from a water surface is determined by the angle of the incident ray, and the index of refraction. This means that light is reflected differently depending on whether a ray is hitting from outside or inside of the water's surface. For water surfaces, we use standard Fresnel term [FvDFH90] for calculating the reflection ratio rather than the approximated Fresnel term for bubble films. As seen in Figure 12, light reflects differently whether hitting a fluid surface from inside or outside, or hitting a bubble film. After calculating the Fresnel term, we render specular highlights as the actual reflections of lights. For the inside of refractive surfaces, approximating the specular highlights with Phong or Blinn shaders for indirect specular highlights is incorrect, as there is no direct pathway for light that is being refracted to a surface point.

While a level set is very powerful at defining numerous shapes, there is little possibility that a level set could be defined so that it sits exactly flush up against an object in the simulation (unless that object is a plane). If the level set is not allowed to overlap objects, there will be a small amount of space between the zero level set (the surface of the fluid) and an object that is in contact with the fluid. Unless this is taken into account, the raytracer will detect two hits, one between the fluid and the empth space and another between the empty space and the object. To avoid this complication, we set up our scene so that the level set overlaps the object when the surface of the fluid is supposed to be against the object.

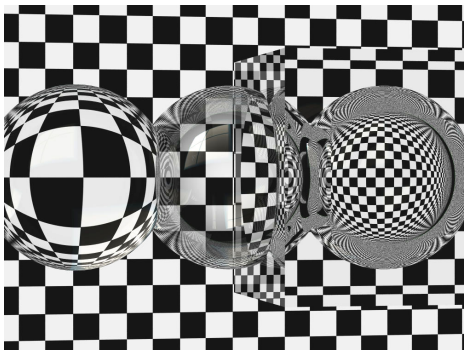


Figure 12: *Refracting water surface.*

From left to right: water sphere, half submerged bubble showing simulated Fresnel reflections on left half, and totally submerged bubble.

Thus, the renderer only has to worry about rendering the boundaries of the object contained in the fluid, not the surface of the fluid that is contained in the object. Bubbles are treated in a similar fashion, as the boundaries of the level set are not rendered when inside of a bubble, but the boundaries of the bubble are rendered when inside of the fluid.

This leads to a hierarchy of objects. Nonfluid/nonbubble objects in the scene get the highest priority. The next priority is bubbles, and the last priority is the fluid. This hierarchy can be seen in Figure 12 and Figure 13. The boundary between a bubble and the surface of the fluid is rendered as the fluid surface. The boundary between a bubble and the air is rendered as a fluid film (approximated Fresnel).

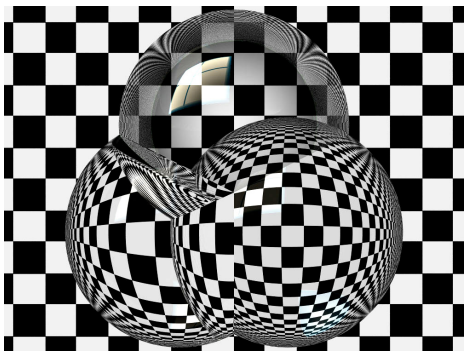


Figure 13: *Hierarchy of surfaces.*

Water (left), glass (right), and bubble (top).

6. Results and Summary

6.1. Issues and concerns

Several assumptions and approximations used in this work lead to certain limitations that we discuss below.

Simulating bubbles that are based on perfect spheres works reasonably well, but there are problems with this approach. In reality, larger bubbles flatten out at the surface so that they resemble domes. Since we can only simulate and render spheres, we avoid creating large bubbles. A similar problem occurs when viewing the bubbles up close, so our method works best for rendering bubbles at a medium distance.

Marker particles may escape occasionally when it is unrealistic for a bubble to form. While we take steps to prevent this, we cannot guarantee that unwanted bubbles will not form. This may not be an issue, because in a production environment, an animator would need the control to add and remove bubbles anyway. In animation software, the fluid surface, velocity and pressure fields, and escaped marker particles could be stored in one pass. Once the desired water motion is obtained, the animator could move on to animating the bubbles, combining our technique with standard particle tools to enhance bubble motion.

Just making the sign of the level set implicit function ϕ negative is not enough to turn all of the air pocket cells into fluid immediately. Adjacent to the air pocket there are near-zero-negative or zero values of ϕ (that are not changed because they are not air). In these regions there may be positive (air) marker particles that have not escaped. These particles may prevent the air pockets from turning entirely into fluid. Since the values of are not negative enough to make these marker particles escape, they persist and continue to contribute to the air pocket.

In our simulation, these particles are removed, but the level set still resists changing to fluid. The equations that reinitialize the level set to a signed distance function avoid changing the values of ϕ as they approach zero. This causes portions of the air pocket to remain air; however, these parts of the level set are continuously moved by the velocity fields and do not persist for more than a few frames. As seen in Figure 14, these residual pieces of air pockets are very tiny. Further, they are obscured by the bubbles that are created by the air pocket and are not noticeable in an animation.

6.2. More realistic fluid?

Our experiments demonstrate that adding bubbles to a fluid simulation can enhance the overall realism of a splashing fluid. In the presence of air, bubbles naturally form in splashing water. Thus, the inclusion of bubbles is both convincing and aesthetically pleasing. Also, our treatment of trapped air pockets gives the more natural appearance of the trapped air turning into bubbles rather than the trapped air suddenly disappearing. Our methods provide an inexpensive way of dealing with air pockets without simulating the air.

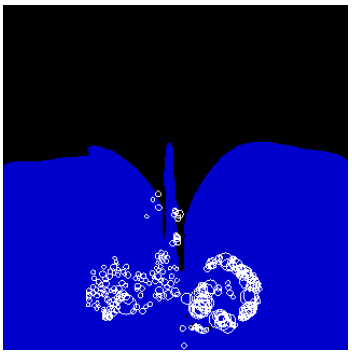


Figure 14: *Small residual pieces of air pocket may persist for a few frames.*

Figures 15a and 15b demonstrate that our method effectively removes air pockets. The air pocket disappears more abruptly than it would if we did not adjust the value of ϕ within the pocket, and the fluid simulation were simply allowed to engulf it. However, the end effect is the same, as the disappearance of air into the fluid looks unrealistic. The creation of bubbles hides the disintegration of air pockets and creates more true to life animations as seen in Figure 15c.

Our experience has been that by creating and rendering bubbles, the added detail distracts from other problems that might otherwise be apparent with the fluid's appearance. Overall, these methods are a good way of creating realistic bubbles if viewed from a medium distance. Figure 16 affirms this observation, showing frames taken from our animations of splashing and sloshing fluids.

References

[BKZ92] BRACKBILL J., KOTHE D., ZEMACH C.: A continuum method for modelling surface tension. *Journal of Computational Physics 100* (1992), 335–354.

[Bol90] BOLTON F.: A computer program for the simulation of two-dimensional foam. *www.tcd.ie/Physics/Foams/plat.html* (1990).

[CJR95] CHEN S., JOHNSON D., RAAD P.: Velocity boundary conditions for the simulation of free surface fluid flow. *J. of Comput. Phys. 25* (1995), 749–778.

[Dur95] DURIAN D.: Foam mechanics at the bubble scale. *Physical Review Letters 75* (1995), 4780–4783.

[Dur97] DURIAN D.: Bubble-scale model of foam

mechanics: Melting, nonlinear behaviour and avalanches. *Physical Review E 55* (1997), 1739–1751.

[EMF02] ENRIGHT D., MARSCHNER S., FEDKIW R.: Animation and rendering of complex water surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH '02) 21*, 3 (2002), 736–744.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. SIGGRAPH '01* (2001), pp. 23–30.

[FM96] FOSTER N., METAXES D.: Realistic animation of liquids. *Graphical Models and Image Processing 58* (1996).

[FvDFH90] FOLEY J. D., VAN DAM A., FEINER S. K., HUGHES J. F.: *Computer Graphics Principles and Practice*. Addison-Wesley, Reading, Mass, 1990.

[Gla00] GLASSNER A.: Andrew glassner's notebook: Soap bubbles, part 2. *IEEE Computer Graphics and Applications 20*, 6 (2000), 99–109.

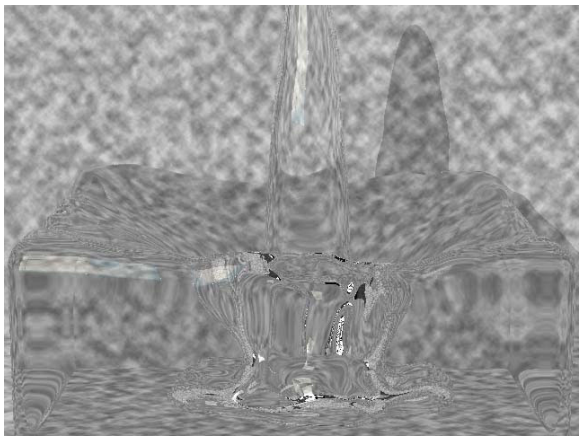
[HK03] HONG J., KIM C.: Animation of bubbles in liquid. In *Proc. Eurographics '03* (2003).

[HW65] HARLOW F., WELCH J.: Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids 8* (1965), 2182–2189.

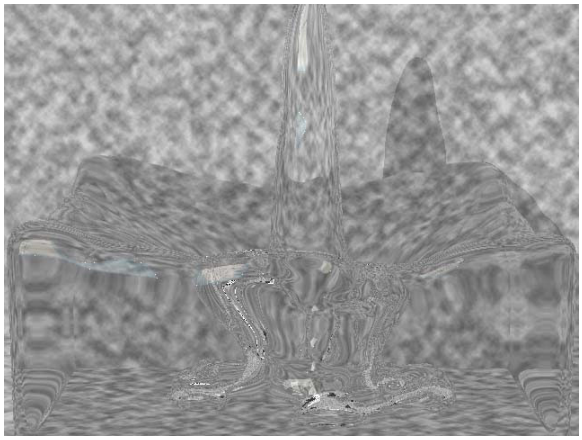
[KVG02] KÜCK H., VOGELGSANG C., GREINER G.: Simulation and rendering of liquid foams. In *Proc. Graphics Interface '02* (2002), pp. 81–88.

[LNS*94] LAFAURIE B., NARDONE C., SCARDOVELLI R., ZALESKI S., ZANETTI G.: Modelling merging and fragmentation in multiphase flows with surfer. *Journal of Compuational Physics 113* (1994), 134–147.

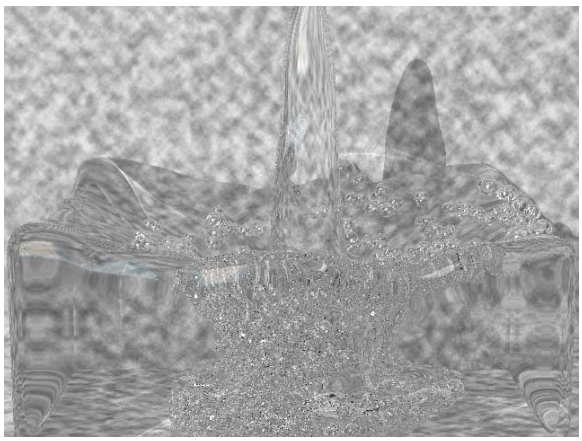
[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH '99* (1999), pp. 121–128.



(a) air pocket before removal



(b) air pocket after removal



(c) air pocket after removal with bubbles

Figure 15: *Animation frame with air pocket*

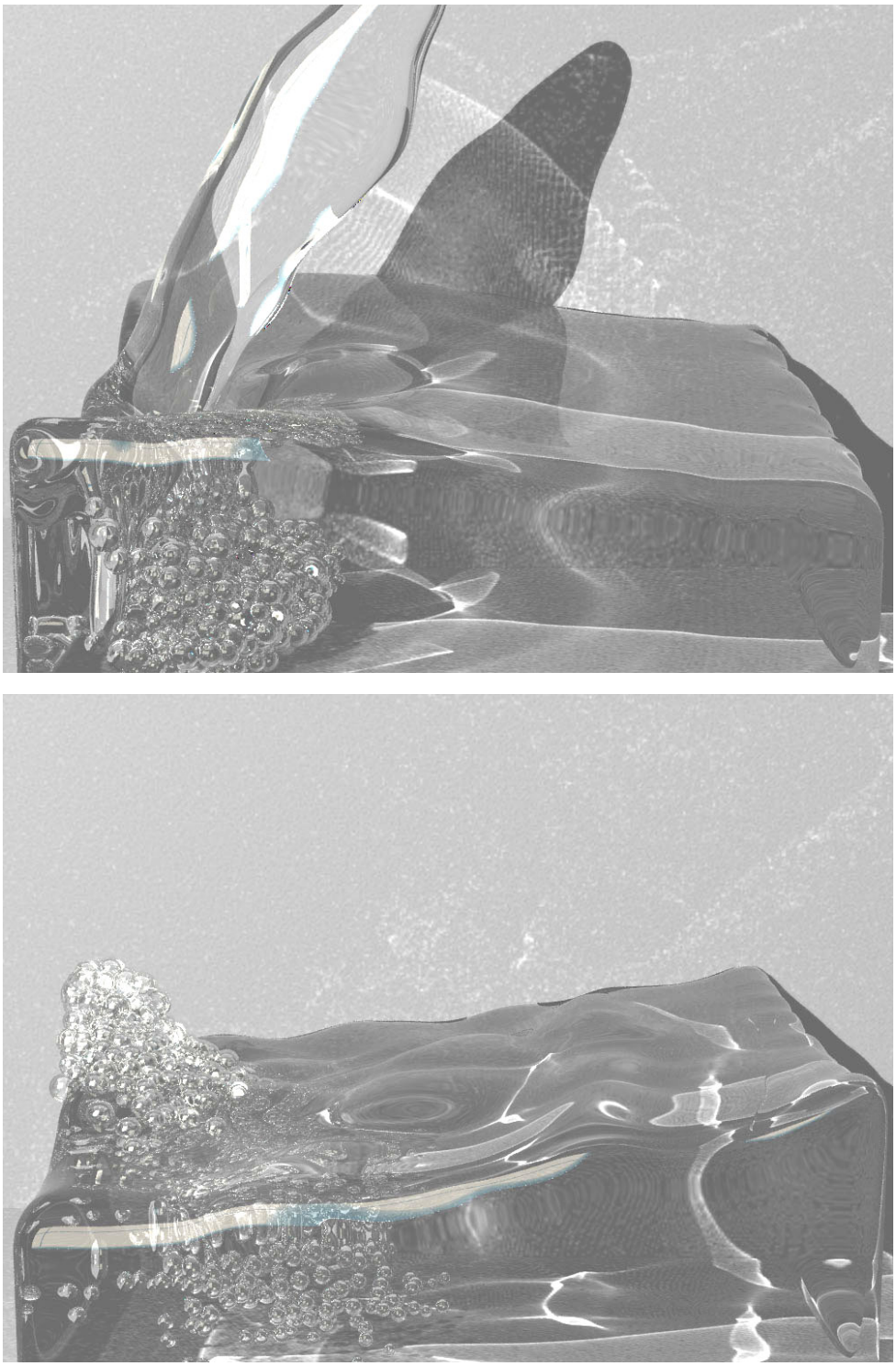


Figure 16: *Fluid splash with bubbles.*