# THE INCORPORATION OF BUBBLES INTO A COMPUTER GRAPHICS FLUID SIMULATION

A Thesis

by

SHANNON THOMAS GREENWOOD

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2004

Major Subject: Visualization Sciences

# THE INCORPORATION OF BUBBLES INTO A COMPUTER GRAPHICS FLUID SIMULATION

A Thesis
by
SHANNON THOMAS GREENWOOD

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

| | |
|---|---|
| Donald House<br>(Chair of Committee) | John Keyser<br>(Member) |
| Frederic Parke<br>(Member) | Phillip J. Tabb<br>(Head of Department) |

May 2004

Major Subject: Visualization Sciences

# ABSTRACT

The Incorporation of Bubbles into a
Computer Graphics Fluid Simulation. (May 2004)
Shannon Thomas Greenwood, B.S., Texas A&M University
Chair of Advisory Committee: Dr. Donald House

We present methods for incorporating bubbles into a photorealistc fluid simulation. Previous methods of fluid simulation in computer graphics do not include bubbles. Our system automatically creates bubbles, which are simulated on top of the fluid simulation. These bubbles are approximated by spheres and are rendered with the fluid to appear as one continuous surface. This enhances the overall realism of the appearance of a splashing fluid for computer graphics.

Our methods leverage the particle level set representation of the fluid surface. We create bubbles from escaped marker particles from the outside to the inside. These marker particles might represent air that has been trapped within the fluid surface. Further, we detect when air is trapped in the fluid and create bubbles within this space. This gives the impression that the air pocket has become bubbles and is an inexpensive way to simulate the air trapped in air pockets.

The results of the simulation are rendered with a raytracer that includes caustics. This allows the creation of photorealistic images. These images support our position that the simple addition of bubbles included in a fluid simulation creates results that are much more true to life.

# TABLE OF CONTENTS

Page

Page

# LIST OF FIGURES

# 1 INTRODUCTION

Water is everywhere. It is what we drink, bathe in, and admire in scenic views. Despite being so common, simulating and rendering water is one of the greatest challenges in computer graphics. Recent methods have produced excellent results but have not achieved complete realism. In part, this is due to the lack of the inclusion of bubbles in a dynamically splashing fluid.

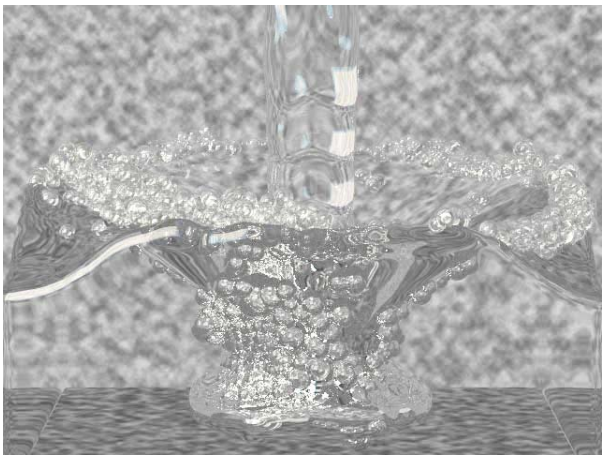

**Figure 1:** *Water simulation with bubbles.*

Bubbles are an inseparable part of water and other fluids. Bubbles are common even in uncarbonated water. Previous methods have not incorporated bubbles and therefore have not looked as realistic as possible. It is our position that the simple addition of bubbles approximated by spheres included in a fluid simulation creates results that are much more true to life (as in Figure 1).

In this thesis, we incorporate bubbles into a fluid simulation using marker particles that have escaped from the outside into the inside of a fluid surface. These escaped marker particles might represent air volume that has been trapped inside the liquid, and are a good indication of where bubbles could form in a dynamically splashing liquid. Further, we examine the possibility of detecting and converting trapped air pockets into bubbles.

We implemented a fluid simulation as described by Enright et al. [6]. On top of this

---

This journal model is IEEE Transactions on Visualization and Computer Graphics.

simulation, we simulated bubbles that are affected by the fluid and each other but have no effect on the fluid. We then render the bubbles and fluid to appear as one fluid surface. The bubble simulation and rendering is largely based on work done by Kück et al. [13].

This thesis is formatted as follows. Section 2 is a brief discussion of background material. The discussion of methodology in this paper is broken up into four sections. In section 3, we discuss the fluid simulation. In section 4, we discuss the creation of bubbles from discarded marker particles. In section 5, we discuss the simulation of these bubbles, and in section 6, we discuss rendering the integrated bubble/fluid simulation. Finally, in section 7, we discuss our results.

# 2 PREVIOUS WORK

## 2.1 OVERVIEW OF PREVIOUS FLUID SIMULATIONS

The simulation of complex water motion for computer graphics, using the Navier-Stokes equations, has been based on results in the computational fluid dynamics community. Foster and Metaxes [10] used the marker and cell method of Harlow and Welch [12] to create a 3D animation of water. Chen, Johnson, and Raad [3] improved this method by placing marker particles only near the surface of the fluid. Stam [14] introduced a semi-Lagrangian treatment of the advection portion of the Navier-Stokes equations allowing large timesteps without causing instability. Foster and Fedkiw [9] introduced a hybrid liquid volume model that combined a level set with marker particles. The markers and level set were advected forward in smaller timesteps than that used for the fluid in order to reduce the error in the representation of the surface. Further work by Enright et al. [6] proposed the use of marker particles outside of the liquid volume in order to better preserve columns of air formed in the fluid.

## 2.2 PREVIOUS WORK FOR FOAM AND BUBBLES

Foams and bubbles have been studied mainly because of their surface minimizing properties. Analytical solutions to bubble clusters up to the size of three bubbles have been found. Glassner modeled these groups with CSG operations.
[11]. Numerical solutions must be used for larger bubble clusters, since their liquid films are not spherical.

Kermode and Bolton simulated foams in two dimensions as a network of curved films [2][15]. Durian created a model that uses a group of interacting bubbles to represent foam [4][5]. 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[13], and our simulation/rendering of bubbles is largely based on these techniques.

# 3 FLUID SIMULATION REVIEW

In order to simulate bubbles with fluids, we had to implement a fluid simulation. We specifically implemented the methods described by Enright et al [6]. This section discusses these methods.

## 3.1 NAVIER-STOKES EQUATIONS

For numerical simulation of the Navier-Stokes equations, the velocities and pressures are typically stored as fields in a three-dimensional grid of cells as in figure 2. The pressures are defined on the center of the cells, and the components of velocities are defined at the centers of the faces of the cells.

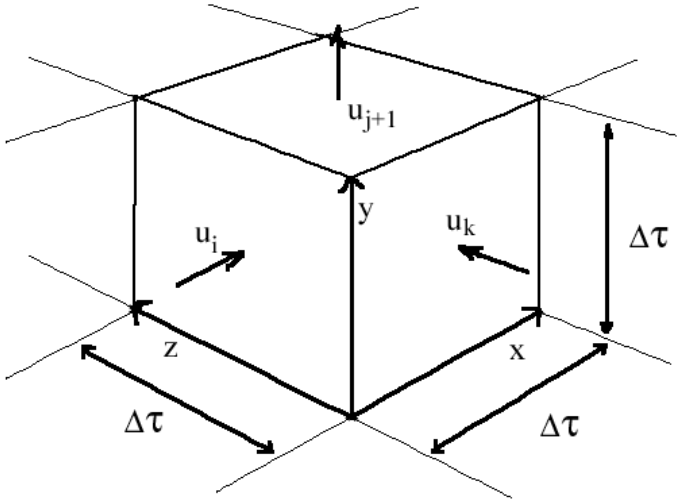


**Figure 2:** *A single grid cell with three of its six face velocities shown* [9].

The Navier-Stokes equations for describing the motion of an incompressible fluid consist of two parts. The first,

$$\nabla \bullet u = 0 \tag{1}$$

enforces incompressibility. Here, u is the velocity field and $\nabla$ is the gradient operator. $\nabla \bullet u$ is the divergence of the velocity, which can be thought of as the net flow of fluid out of a differential cell. By requiring the divergence to be zero, the flow in matches the flow out, and thus mass is conserved.

The second part of the Navier-Stokes equations,

$$u_t = \nu\nabla\bullet(\nabla u) - (u\bullet\nabla)u - \frac{1}{\rho}\nabla p + g \tag{2}$$

couples the velocity and pressure fields relating them through momentum. Here $\nu$ is viscosity, $\rho$ is density, $p$ designates pressure, and $g$ represents external accelerations, like gravity, acting on the fluid. Since there are no analytic solutions, these equations are solved numerically over time to model the behavior of an incompressible liquid.

### 3.1.1 Solving the Navier-Stokes Equations in Multiple Parts

For computer graphics, as long as the fluid looks convincing, it does not matter whether the underlying fluid velocities are completely accurate. This might be quite different for a mechanical engineer who demands accurate fluid velocities and pressures in a simulation of a dam. He is solely interested in achieving accuracy, while in computer graphics we are interested in the creation of attractive images and can ignore accuracy to a certain degree. The methods preferred for computer graphics sacrifice some accuracy for stability and speed. These methods guarantee that the simulation is stable, and allow the use of large timesteps. By allowing large timesteps, computation time is reduced, and the simulation can be executed faster.

As first shown by Stam [14], the Navier-Stokes equations can be solved in four parts. The steps are:

$$w_0(x) \overset{\text{add force}}{\rightarrow} w_1(x) \overset{\text{advect}}{\rightarrow} w_2(x) \overset{\text{diffuse}}{\rightarrow} w_3(x) \overset{\text{project}}{\rightarrow} w_4(x). \tag{3}$$

Here $x$ denotes position. $w_0(x)$ is the velocity at its initial state during a time step. Successive portions of the Navier-Stokes equations are applied creating $w_1(x)$, $w_2(x)$, and $w_3(x)$. The divergent velocity field $w_3(x)$ is finally projected creating $w_4(x)$ which is the divergence free velocity field at the end of the time step (see subsection 3.1.1.4). It should be noted that when explicit methods are used, a simple Euler step is all that is necessary. The computation and

memory requirements for higher order methods make them undesirable when accuracy is not the highest priority.

#### 3.1.1.1 Forces

For the forces part of the equation (g), we simply take an Euler timestep to add in the forces, as in

$$w_1(x) = w_0(x) + g\Delta t\,. \tag{4}$$

#### 3.1.1.2 Advection

For the advective[1] term of the equations ($-(\mathrm{u}\bullet\nabla)\mathrm{u}$), a semi-Lagrangian method is used to enforce stability, as in

$$w_2(x) = w_1(p(x,-\Delta t))\,. \tag{5}$$

The velocity is traced back in time one timestep, and the backtraced velocity is used for the new velocity field. The tracing function shown here, $p(x,-\Delta t)$ takes two parameters. The first is the position $x$. The second parameter $-\Delta t$ tells the tracing function to backtrack one timestep. The result returned is the position traced from the position $x$ one timestep backward through the velocity field $w_1(x)$. The velocity value of $w_1(x)$ at this position becomes the values of $w_2(x)$ at position $x$. Stam[14] implemented the backtracing function $p(x,-\Delta t)$ with a second order adaptive particle tracer. Higher order interpolants are undesirable due to high overshoots and oscillations. We simply used a small Euler step to backtrace velocities. This may be inefficient, but is not a problem because this part of the Navier-Stokes equation is not computationally expensive.

By backtracing the velocities, the momentum of the velocity field is carried forward in time. This method guarantees stability because no velocity values can be higher than previous velocities.

---

[1] Advection and Convection are used here as in [13]. In fluid dynamics, advection and convection are often used interchangeably to describe fluid movement.

3.1.1.3 Diffusion

For the convective[1], diffusion, or viscous term ( $\nu\nabla\bullet(\nabla u)$ ), Stam used an implicit solution. We explicitly solved this portion of the equation with local viscosity adjustment as suggested by Foster and Fedkiw [9] where

$$w_{3(x)} = w_2(x) + \Delta t \frac{\partial w_2}{\partial t} \quad , \tag{6}$$

and

$$\frac{\partial w_2}{\partial t} = \nu\nabla^2 w_2. \tag{7}$$

$\nabla^2 w_2$ is discretized using standard central differencing and if $\frac{\partial w_2}{\partial t}$ is too large causing instability, it is clipped to a manageable value (based on the CFL condition, see section below).

3.1.1.4 Projection

Finally, incompressibility is enforced. Here, projection is defined as the operation that begins with the divergent velocity field $w_3$ and results in a divergence free velocity field $w_4$ (using the pressure field $p$ ). As discussed by Stam[14], a single equation relating velocity and pressure can be obtained by combining eq.1 and eq.2. A mathematical result known as the Helmholtz-Hodge Decomposition, states that a vector field can be divided into a divergence free vector field and the gradient of a scalar field. This takes the form

$$w = u + \nabla q, \tag{8}$$

where $u$ is the divergence free vector field ( $\nabla\bullet u = 0$ ) and q is the scalar field. This allows the definition of a projection operator P which projects any vector field $w$ onto its divergent free part $u = Pw$.

This operator is implicitly defined by taking the divergence of both sides of the equation giving

$$\nabla\bullet w = \nabla^2 q\,. \tag{9}$$

This causes $u$ to drop out of the equation because $\nabla\bullet u = 0$. We can solve for $q$ if we have w. Rearranging equation 8 gives

$$u = Pw = w - \nabla q\,. \tag{10}$$

To apply this to the Navier-Stokes equations, we substitute the divergent velocity field $w3$ and the scalar field $\frac{1}{\rho}\nabla p$ into equation 9 and we get

$$\nabla^2 p = \rho \nabla \bullet w_3 . \tag{11}$$

After solving for pressure, we can solve for the divergence free velocity $w_4$ by substituting into equation 10 giving

$$w_4 = w_3 - \frac{1}{\rho}\nabla p . \tag{12}$$

The laplacian operator ($\nabla^2$) on the left side of equation 11 is a matrix with length and width of N, where N is the number of cells in the simulation grid. To solve for $p$, this matrix must be inverted and is too large to be inverted by analytical methods. Fortunately, the matrix is mostly empty since each cell is only related to adjacent cells. A sparse matrix solver may be used to numerically invert the matrix to a high degree of accuracy. Foster and Fedkiw [9] used a preconditioned conjugate gradient method in order to invert the matrix and solve for pressure.

It should be noted that as in equation 11, the gradient of the velocities is needed at the location of the pressure, and the gradient of the pressure is needed at the location of the velocities. Thus by staggering velocities and pressures fields, as in Figure 2, there is better accuracy than defining all values at the center of the cell.

### 3.1.2 Empty Cells

The Navier-Stokes equations solve for the velocities inside of the fluid, but they do not specify the velocities outside of the fluid or at the boundary of the fluid. Cells that contain only fluid are considered fluid cells, and they are only governed by the Navier-Stokes equations. Cells that contain no fluid are considered empty cells and are not simulated. Not simulating the air outside of the fluid saves computation time. Since the air outside of a fluid often does not affect the fluid's motion, this is a reasonable time saving measure.

### 3.1.3 Surface Cells

Cells that contain both air and fluid are considered surface cells. The pressures of the surface cells are set to a user defined atmospheric constant. The velocities of the surface cells

must be explicitly set to be divergence free. The setting of surface velocities is a heuristic, and its main purpose is to keep the fluid from unrealistically gaining or losing fluid. The primary movement of the fluid is government by the Navier-Stokes equations in the fluid cells. Chen et al. [3] discusses individual cases for making the surface cells divergence free. Enright et al. [6] discusses using the methods described by Adalstein and Sethian [1] to extrapolate velocities for surface cells into a few empty cells while simultaneously making the surface cells divergence free.

It is useful to extrapolate velocities into empty cells adjacent to the surface because of the semi-Lagrangian method for solving the advection term. Backtracing velocity values may lead out of the current fluid. If velocity values are not extrapolated into empty cells, the velocities in the empty cells are zero. Backtracing zero values can cause the fluid to settle faster.

### 3.1.4 CFL Condition

The goal of solving the Navier-Stokes equations with these methods is stability, so the largest timestep possible can be used. The CFL condition (Courant-Friedrichs-Levy) states that the timestep must be smaller than the minimum time over which something significant can happen. Using the methods outlined above, the Navier-Stokes have been solved with good results at 5 times the CFL condition [9]. For this discrete system describing fluid, the relevant CFL condition is the time in which the fluid can jump over an entire cell skipping it entirely, or

$$\Delta t < \Delta\tau / |u|, \tag{13}$$

i.e. the timestep should be less than the width $\Delta\tau$ of a cell divided by the magnitude of the maximum velocity. If it is possible for fluid to skip over an entire cell, velocities in the skipped cell may not affect portions of the fluid. This is significant because the fluid passes over the cell without being affected by the velocity within the cell.

## 3.2 LEVEL SET

We used a level set method to represent the fluid as described in [6] and [9]. A level set is a temporally smoothed dynamic implicit function. The function is defined on a high resolution Eulerian sub-grid that sits inside of the Navier-Stokes grid. The isocontour where the implicit function $\phi$ equals zero defines the surface interface. $\phi$ is a signed distance function that is

positive outside of the surface and negative inside. The surface can be ray traced directly using a root finding algorithm. The zero values can be found easily because at any point the value of $\phi$ explicitly gives the minimum distance to the surface.

The level set is moved forward in time by

$$\phi_t + u \bullet \nabla \phi = 0, \tag{14}$$

which is similar to the advection term of the Navier-Stokes equations. This advection equation could be solved using semi-Lagrangian methods, but these are too inaccurate. To achieve high accuracy, we use the higher order upwind differencing procedure described in [7]. This uses a wide envelope when discretizing $\nabla \phi$ to achieve high-order spatial accuracy.

As the level set is advected, it is stretched and compacted. A signed distance function must be maintained for efficient rendering and accurate simulation.

### 3.3 PARTICLES

Level sets do not preserve sharp detail for coarse grids and dynamic surfaces, so there may be severe volume loss as seen in Figure 3. To address this issue, Foster and Fedkiw [9] combined a level set with particles on the inside of the fluid.

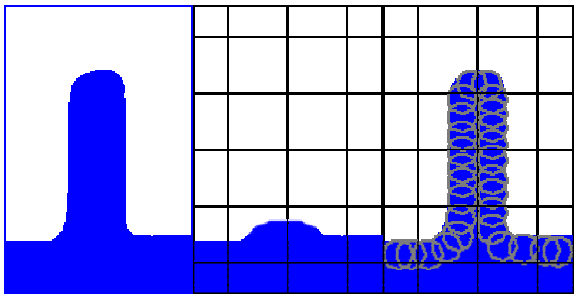

**Figure 3:** *Prevention of volume loss using marker particles on the inside of a level set. Thin water column (left), representation by level set alone (middle), and level set with marker particles on the inside(right).*

While this prevented volume loss in sharp areas of the level set, it did not prevent volume gain in sharp areas that curved inward into the fluid surface such as an air column as seen Figure 4.

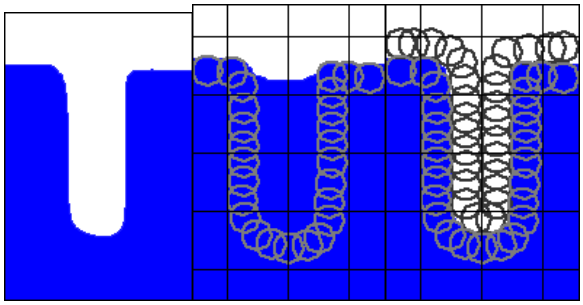

**Figure 4:** *Particles on the inside only versus particles on both sides. Thin air column(left), hybrid representation with particles on inside only (middle), and hybrid representation with particles on inside and outside(right).*

Enright et. Al [6] used particles on the inside and outside to prevent both volume loss and volume gain of the level set. It should be noted that when using particles alone to represent a fluid, it is very difficult to maintain an appealing surface, so a hybrid level-set/particle method is the optimal solution for representing the fluid surface.

### 3.3.1 Particle Error Correction

We associate a spherical implicit surface function

$$\phi_p(\vec{x}) = s_p(r_p - |\vec{x} - \vec{x}_p|) \tag{15}$$

with each particle, where $s_p$ is the sign of the particle (negative inside, positive outside), $r_p$ is the radius, and $x_p$ is the position[6].

When we reconstruct the level set, we compare $\phi$ with $\phi_p$ at the grid points containing the particle. For a negative particle we update $\phi$ with $\phi_p$ if $\phi_p$ is the lower value. For a positive particle we do the same if $\phi_p$ is the greater value.

The marker particles are integrated forward separately from the level set using a forward Euler time integration scheme. Particles that are on the wrong side of the interface by more than their radius are used to reconstruct the level set. For moving the particles and level set, a smaller timestep obeying the CFL condition (see equation 13) should be used to avoid dissipation [9]. This is not a problem because updating the surface is not too expensive computationally. Enforcing incompressibility is the most expensive part of the algorithm.

### 3.3.2 Particle Seeding and Reseeding

Before the simulation, particles are seeded on both sides of the interface. Particles are randomly placed in the desired density per cell. Enright et al. [6] seeded 32 to 64 particles per cell. Having more marker particles per cell than 64 has little effect because there is thorough coverage. Having fewer marker particles per cell may not provide enough coverage and may allow volume loss/gain of the level set.

Enright et al. [6] seeded particles three cells on each side of the interface. This distance was chosen because it works well. Particles are not needed further from the interface because they will not be used to reconstruct the surface. Seeding particles over a smaller distance from the interface might not provide enough coverage.

Over time, the interface tears and stretches so that the original particle density may not be maintained. Particles need to be reseeded every so often to maintain the desired density (Enright et al. reseeded the particles every 20 frames). Particles that are further than the appropriate distance from the interface are removed from the simulation. The reseeding operation should avoid replacing particles that are overlapping or adjacent to the interface. These particles contain very specific information about the fluid surface If necessary, particles that are further from the interface should be removed to reduce the particle density in a cell.

### 3.3.3 Particle Radius Adjustment

The radius dynamically changes as the particles are moved relative to the surface as seen in Figure 5.

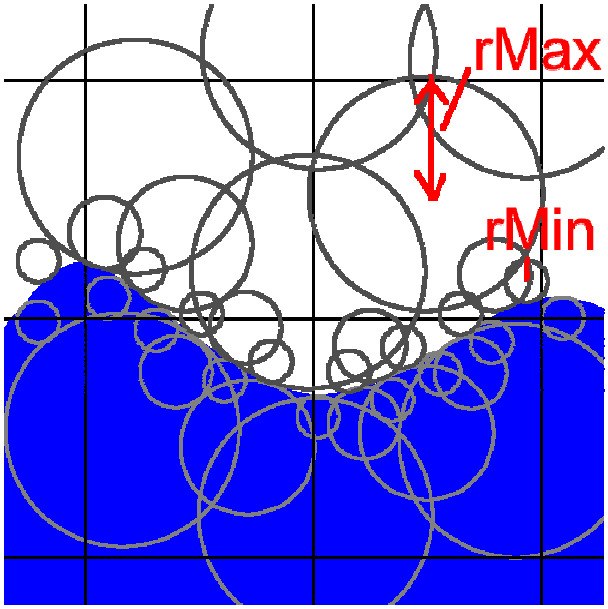


**Figure 5:** *Particle radii set by their distance from the fluid surface.*

The radius is set by

$$r_p = \begin{cases} r_{\max} & \text{if } s_p\phi(\overrightarrow{x_p}) > r_{\max} \\ s_p\phi(\overrightarrow{x_p}) & \text{if } r_{\min} \le s_p\phi(\overrightarrow{x_p}) \le r_{\max} \\ r_{\min} & \text{if } s_p\phi(\overrightarrow{x_p}) < r_{\min} \end{cases}, \tag{16}$$

where $r_{\min}$ and $r_{\max}$ are the minimum and maximum radii that are assigned to particles. Enright et al. [6] used .1 times the width of a cell as the minimum radius and .5 times the width of a cell for the maximum radius (which were chosen because they work well). The radius adjustment puts the edge of the spherical particle flush with the zero level set when possible. This helps makes the fluid surface smooth.

### 3.3.4 Escaped Particles

Even though a particle's volume may be jutting over onto the other side of the level set, we only apply error correction once a particle is at least it's own radius on the wrong side of the interface. As seen in Figure 6, a particle must be entirely on the wrong side of the interface for error correction to be applied for that particle.

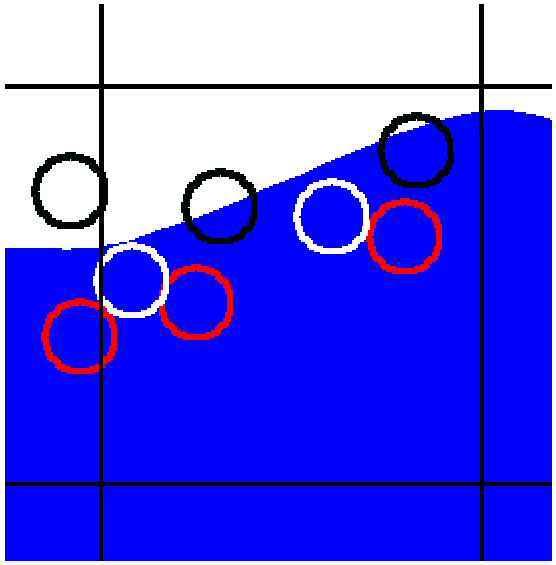

**Figure 6:** *Error correction of positive particles. Particles not used in error correction (black), particles used in error correction (white), and escaped particles (red).*

This is done because we only want to apply error correction when there is a large disparity between the level set and the particles. Otherwise, the smoothness of the level set is the priority.

These corrections need to be done after the level set and particles are integrated forward, and after the level set is reinitialized to a signed distance function. The radii of the particles are adjusted after the second correction step. Particles that are 1.5 times their radius on the wrong side of the surface may prevent the surface from being smooth (this is an arbitrary choice that works well).

Foster and Fedkiw [9] suggested that particles that have escaped to the outside of the surface could be rendered as spheres simulating drops. The particles that escape from the outside to the inside were just deleted in previous fluid simulations. In the next section, we will explore the use of these escaped particles to create bubbles.

# 4 CREATING BUBBLES FROM ESCAPED MARKER PARTICLES

In the real world, bubbles are created whenever air is trapped inside of a fluid. Bubbles persist at the surface because of the surface tension in the film and the pressure within the bubble.

Surface tension effects can be added to fluid simulations. The air can be simulated as a second fluid, and bubbles can be simulated 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 gird.

For our simulation, we create bubble objects that are moved by the velocities and pressures within the fluid simulation. Our bubbles are passive, and have no effect whatsoever 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 on the surface of the fluid. In reality, bubbles moving through a fluid do affect the motion of the fluid. Since a bubble is part of the fluid, when a bubble moves, what one sees is the motion of the surface of the fluid. Since we are representing this 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).

At the surface, a single bubble has miniscule mass compared to the water itself, so it will not cause noticeable motion if the water is moving. If the water is still, a single bubble could cause noticeable ripples, since that bubble causes the only motion in the water. A still fluid is not the problem that we are interested in anyway, because for a still fluid, the Navier-Stokes equations are already excessive.

Using passive bubbles is simpler and is more practical than the alternative of allowing the bubbles to create forces or directly manipulate fluid velocities. Once the fluid simulation is run and saved, the bubble simulation can be tweaked to the needs of the animator. For example, bubbles could be added and removed as desired. If the bubbles affected the fluid, the entire fluid simulation might need to be re-run each time a bubble is changed. Since the fluid simulation takes much longer than the bubble simulation, that is undesirable.

## 4.1 BUBBLE CREATION

In this thesis, we explore the idea of creating bubbles from escaped marker particles. 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.

We simulated the fluid as specified by Enright et al. [6]. The bubble simulation created is independent of how the Navier-Stokes equations are solved, but is dependent on the hybrid-particle method representation of a fluid surface. There have to be marker particles representing the outside of the fluid in order to use them to create air bubbles.

## 4.2 TREATMENT OF AIR POCKETS

While Enright et al. [6] improved the representation of the fluid surface so that the level set did not leak into surrounding air columns, they still did not simulate the air. By not simulating air as an incompressible or compressible fluid, 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.

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 [8]. The cells are treated like pixels, and cells with positive $\phi$ are "painted" as one "color" which designates them as air cells. All other cells including wall cells are "painted" another "color" designating that they are not air cells. 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). The empty cells are "painted" a "color" that designates that these cells are connected to the atmosphere. After the fill algorithm is complete, empty cells that are not "painted" the atmosphere "color" are parts of air pockets.

The only modification needed is that diagonal cells from an atmosphere cell should be included only if the value of $\phi$ between the cells is positive. If the value of $\phi$ at the junction

between the cells is positive, the air extends diagonally between cells. Otherwise, the diagonal of the cell is blocked off with fluid as seen in Figure 7.

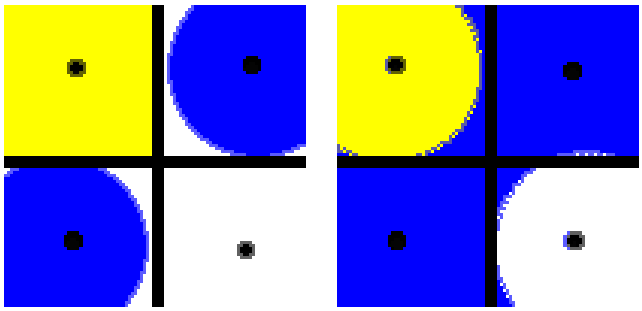

**Figure 7:** *Diagonal inclusion cases. Amosphere cells are marked in yellow. The fluid is marked in blue. Case when diagonal cell should be included with atmosphere (left) and when it should not (right).*

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 changing the value of $\phi$ to a negative value. Since the marker particles within these cells are now far from the interface, they are considered escaped and become bubbles. This gives the appearance that the air pocket changes into bubbles. The velocities of the cells in the air pockets will be initialized to a reasonable value through the extrapolation of surface velocities (in the surface conservation step).

## 4.3 AVOIDING UNREALISTIC BUBBLES

Since, we are creating bubbles from escaped marker particles, we want to avoid creating bubbles in cases where it is not feasible for bubbles to form. One precaution taken is to create a bubble only if the curvature of $\phi$ is negative at the position of the particle. If there is a positive curvature, the water is curved outward towards the air, and bubbles should not form. If there is a negative curvature, the air is curving toward the water and it is feasible that bubbles could form.

### 4.3.1 Small Particles

By the Enright et al method [6], particles' radii are adjusted in order to keep the surface smooth. It is common for the level set to differ from the particles by enough to cause particles of the minimum radius to escape. In many of these situations, it is not appropriate for bubbles to

form, so we want to use caution when using these small particles to create bubbles. Since a particle with larger than the minimum radius represents a larger move of the surface, we accept these particles as a basis to create bubbles. If a bubble is created from a particle with the minimum radius, we mark that that is the case. Once all of the bubbles are created, we check that new bubbles created from small particles are in contact with new bubbles created from larger particles. If there is no contact, then we remove the bubble created from a small particle.

### 4.4 BUBBLE SIZE

In our method, the bubbles are based on the position and radius of the escaped marker particles. Since the radius of the particle is between .1 and .5 of the cell width, creating bubbles directly from the size of the radius would create extremely tiny bubbles for fine grids. If the escaped particle's radius were just scaled up by a constant value, there would not be enough variation in bubble size. Even when checking validity, a large proportion of particles with the minimum radius escape. Also, when air pockets form, all of the marker particles are of the maximum size away from the surface of the air pocket.

With just a constant scale, there would be too many bubbles with identical radius. Bubbles radii could be generated totally randomly, but there is valid information in the marker particles' radii to be used. 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 determined by the radius of the marker particle. Particles with the maximum radius create bubbles with one average and std. deviation. Particles with the minimum radius create bubbles with a different average and std. deviation. For particles with radius sizes in between the minimum and maximum, we used interpolated values of the averages and standard deviations. Values that are too large or small are recalculated until they are 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 8, this method is independent of the cell width and grid size, so that similar bubbles will be created for different grid resolutions.

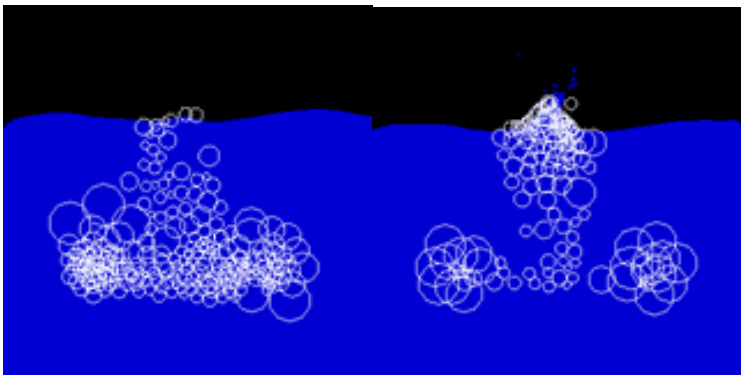

**Figure 8:** *Bubbles created in different grid resolutions. 30x30 grid (left) and 60x60 grid (right).*

## 4.5 BUBBLE MERGING AND POPPING

It should be noted that adjacent marker particles overlap. Further, if the bubbles are larger than the original marker particles, then there is 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 not to implement this, since we model bubbles with spheres, larger bubbles are undesirable. The limitations of large bubbles are discussed in the last section. We did remove bubbles that were completely encompassed by other bubbles.

A bubble pops when one of its films drains too much and breaks. As discussed in Kück et al[13], 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. Thus, 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 parameterized this behavior by having a lifetime of a small bubble size, and scaled down its lifetime by a scale associated with a larger bubble size as follows

$$L = \begin{Bmatrix} 0 & \text{if } r_b \le r_s \\ \dfrac{(r_b - r_s)}{(r_l - r_s)}\ell & \text{if } r_b > r_s \end{Bmatrix}, \tag{17}$$

where $r_b$ is the bubble's radius, $r_s$ is the parameter of the radius where the bubble's lifetime is not scaled down, and $r_l$ is parameter of the radius associated with the scale $\ell$. Then, at each timestep in the simulation it is checked whether

$$\mho < \frac{(1+L)}{T}\Delta t, \tag{18}$$

where $\mho$ is a uniformly distributed random value between zero and one, $T$ is the parameter for bubble lifetime, and $\Delta t$ is the timestep. If this condition is true, the bubble is removed.

We allow $L$ to reach values greater than one to allow increasingly short lifetimes for large bubbles. $L$ does not have to be clipped to 0, as bubbles smaller than the reference bubble might have longer average lifetimes. It is arbitrarily clipped to 0 to keep it from reaching -1 where it would cause infinite bubble lifetimes.

We modeled the behavior of bubbles in contact with obstacles lasting longer by having a separate lifetime parameter for bubbles in contact with walls.

The film that ruptures could be the film between bubbles, and the result of its breakage is the merging of bubbles. This type of merging behavior was not implemented because, as mentioned before, we tried to avoid creating large bubbles.

Also, it should be noted that bubbles should only be removed once they reach the surface. It would not be realistic for bubbles to "pop" while inside of the fluid.

### 4.6 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 volume for the same number of marker particles per cell. This is a problem as the larger grid may create more bubbles than the smaller grid. It is useful to test bubble parameters with smaller grid sizes, so consistency is important when changing to larger grids. A marker/volume density is chosen in which all escaped particles are considered to create bubbles. In the step

where bubbles are created, the desired bubble marker density is divided by the actual marker/volume density. If a random number is below this threshold, then a bubble is considered for creation (it still may be rejected based on curvature or maker particle radius). That means that for a simulation with 8 times the marker particles per volume density of the desired value, we only use one eighth of the escaped marker particles to create bubbles.

# 5 BUBBLE SIMULATION

## 5.1 SIMULATION OF FOAMS FROM KÜCK ET AL. [13]

We based our foam simulation on the methods outlined by Kück et al. [13], and this entire subsection discusses only these methods. The exact locations of foam films are not simulated. Bubbles are simulated by spheres of fixed radii and are moved according to assumed forces.

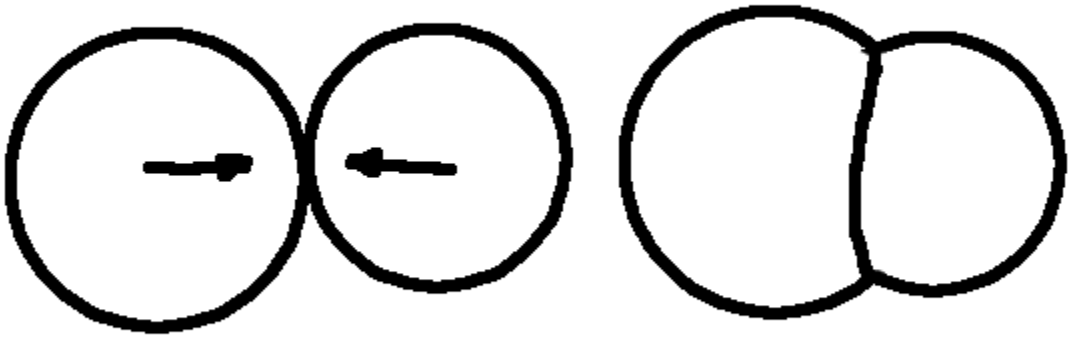

**Figure 9:** *Attractive forces acting on touching bubbles.*

Attractive and repelling spring forces are created to cause the bubbles to overlap and appear to be part of a foam structure. The desired effect is that the forces will cancel each other out when the bubbles overlap the desired amount.

The repelling force $F_{ij}^{r}$ (the force on bubble i due to contact with bubble j) is modeled with a spring of rest length

$$l_{ij} = r_i + r_j \,, \tag{19}$$

where $r_i$ and $r_j$ are the radii of bubbles i and j respectively. The equation for the spring force is then

$$F_{ij}^{r} = k_r \left( \frac{1}{\left\| p_i - p_j \right\|} - \frac{1}{l_{ij}} \right) (p_i - p_j), \tag{20}$$

where $k_r$ is a user defined coefficient that determines how strong the repelling behavior of the bubbles is, and $p_i$ and $p_j$ are the positions of the spheres representing the bubbles i and j.

The attractive forces depend not only on the distance between bubbles, but on how many bubbles each bubble is in contact with. More specifically,

$$F_{ij}^{a} = k_a c_{nb} c_{dist} \frac{p_j - p_i}{\left\| p_j - p_i \right\|} \tag{21}$$

with

$$c_{nb} = \frac{\frac{1}{\left| NB_i \right|} + \frac{1}{\left| NB_j \right|}}{2} \tag{22}$$

and

$$c_{dist} = \frac{\left\| p_j - p_i \right\| - \max(r_i - r_j)}{\min(r_i, r_j)}. \tag{23}$$

$\left| NB_i \right|$ is the number of spheres overlapping with sphere i in the current configuration. $k_a$ is a user-defined term that determines how strong the attractive force between bubbles is. $c_{nb}$ causes the attraction force to be smaller for bubbles in larger clusters. $c_{dist}$ becomes zero when the center of the smaller bubble rests on the edge of the larger bubble, and $c_{dist}$ increases when the distance is larger. It should also be noted that $c_{dist}$ becomes less than zero when the larger bubble encompasses the center of the smaller bubble. This causes the attraction force to be a repelling force preventing bubbles from overlapping too much.

This viscosity between bubbles as they move with respect to their neighbors is modeled by

$$F_i^v = k_v(\overline{v_i} - v_i) \tag{24}$$

$$= k_v\left(\left[\sum_{j\in NB_i}\frac{v_j}{|NB_i|}\right] - v_i\right), \tag{25}$$

where $k_v$ is the user defined parameter for the viscosity, $v_i$ is the velocity of the bubble i, and $\overline{v_i}$ is the mean velocity of the bubbles in contact with bubble i.

Also, Kück et al. takes into account friction between a bubble and an obstacle (that the bubble is in contact with) by

$$F_i^{of} = k_{of}(\overline{v_i^o} - v_i), \tag{26}$$

where $k_{of}$ is the user defined friction coefficient and $\overline{v_i^o}$ is the average velocity of obstacles in contact with bubble i. A constant force $F_g$ models gravity and acts on all bubbles. Also, air resistance is modeled with

$$F_i^{air} = -k_{air}v_i, \tag{27}$$

where $k_{air}$ is the user-defined parameter for air resistance.

In order to solve for velocity, Kück et al. grouped forces. The force groupings are given by

$$F_i^{total} = F_i^{ra} + F_i^{fr} + F^g, \tag{28}$$

with

$$F_i^{ra} = \sum_{j\in NB_i}(F_{ij}^r + F_{ij}^a) + \sum_{k\in NO_i}(F_{ik}^{or} + F_{ik}^{oa}); \tag{29}$$

$$F_i^{fr} = F_i^v + F_i^{of} + F_i^{air}, \tag{30}$$

where $F_i^{total}$ is the total force acting on bubble i, $F_i^{ra}$ is the grouping of all attractive and repelling forces with adjacent bubbles( $NB_i$ ) and objects ( $NO_i$ ), and $F_i^{fr}$ represents all friction effects on bubble i with other bubbles, obstacles, and the air. It also should be noted that $F_{ik}^{or}$ is the attractive force on bubble i due to object k, and $F_{ik}^{oa}$ is the repelling force on bubble i due to object k. These are treated similarly to bubble-to-bubble attraction and repulsion forces.

In order to solve for velocity, Kück et al. assumed that the bubbles are massless. This means that the sum of all forces $F_i^{total}$ is zero, and allows a direct solution for velocity. By combining equations 25 through 30 we get the explicit solution giving

$$v_i = \frac{k_v \overline{v_i} + k_{of} \overline{v_i^o} + F_i^{ra} + F^g}{k_v + k_{of} + k_{air}}. \tag{31}$$

This is a first order ODE system. It should be noted that the spring forces are dependent on which bubbles overlap. This information must be updated for each timestep. The result of the preceding timestep is used as the configuration for the timestep.

To incorporate the bubble simulation model from Kück et al. [13] to work with fluids, we have to add forces from the fluid to the bubble simulation.

## 5.2 SIMULATION OF BUBBLES WITH FLUID

### 5.2.1 Pressure Forces on Bubbles

Bubbles within the fluid are subject to a force due to pressure. Foster and Metaxes [10] modeled buoyant objects that were subject to a force related to the negative gradient of the pressure times the volume. We created the pressure force

$$F_i^p = -K_p \nabla p_i V_i, \tag{32}$$

where $\nabla p_i$ is the gradient of the pressure at the position of bubble i, $V_i$ is the volume of bubble i, and $K_p$ is a user defined animation parameter that can adjust the contribution of pressure on a bubble's motion. Since our forces such as friction are modeled linearly, for large bubbles, velocities can get unrealistically large. For the calculation of $F_i^p$ we clipped the value of $V_i$ to a user defined limit to keep bubbles from moving unrealistically fast.

### 5.2.2 Fluid Viscosity and Bubbles

The viscosity of the fluid affects a bubble's velocity. This is modeled by

$$F_i^{visc} = K_{visc}(v_i^{fluid} - v_i), \tag{33}$$

where $F_i^{visc}$ is the force on bubble i due to the viscosity of the fluid, $v_i^{fluid}$ is the velocity of the

fluid at the position of bubble i, and $v_i$ is the velocity of bubble i. $K_{visc}$ is a user-defined coefficient of viscosity. $K_{visc}$ could be the same as the viscosity coefficient used in the Navier-Stokes equations; however, we used a separate parameter for more control over the animation.

## 5.3 BUBBLE FORCES IN DIFFERENT AREAS

As in previous work [13], we make the assumption that the bubbles are massless to solve for their velocities. Depending on where the bubble is in the simulation, there is a different contribution of different forces.

The four areas that we defined that a bubble can be in are below, adjacent below, adjacent above, and above the surface as seen in Figure 10.

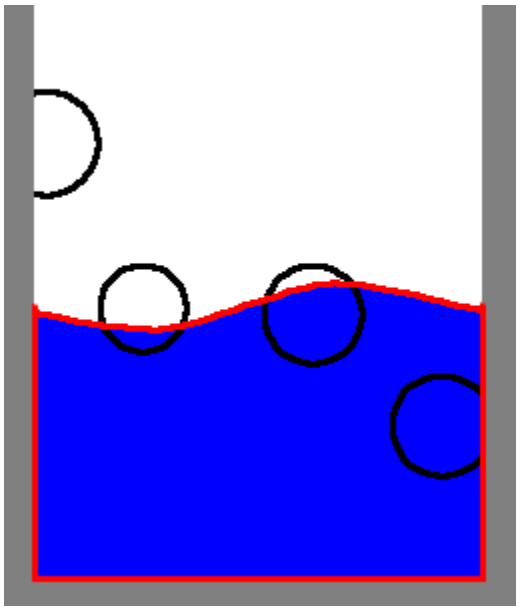

**Figure 10:** *Bubble regions. Above the surface (left), adjacent above (middle left), adjacent below (middle right), and below the surface (right). Fluid surface interface is marked in red.*

### 5.3.1 Bubble Forces Below the Surface

In wet foams there is only a repulsion force between bubbles [13], so we did not model attractive forces between bubbles when they are totally under the surface of the fluid. Also, under the surface of the fluid, we did not have a friction force between bubbles. We did not want bubbles beneath the surface to remain in contact since they only repel each other. Also, gravity

will not have significant impact, as the pressure force will dominate. In order to solve for velocity, we combine equations 32 , 33, and 35 into one equation giving

$$F_i^{total} = F_i^r + F_i^p + F_i^{visc}, \tag{34}$$

with

$$F_i^r = \sum_{j \in NB_i} (F_{ij}^r) + \sum_{k \in NO_i} (F_{ik}^{or}). \tag{35}$$

$F_i^r$ is the grouping of repelling forces between bubbles. We make a massless assumption like Kück et al. [13] used in the derivations of equation 31. By setting $F_i^{total}$ to zero and solving, we get

$$v_i = \frac{1}{K_{visc}}(K_{visc} v_{fluid} + F_i^r + F^p), \tag{36}$$

where the only contributions are from viscosity with the fluid, repulsion forces between bubbles and other objects, and the force due to pressure.

It should be noted that we omitted including a friction force between bubbles and object as done by Kück et al. [13]. It did not seem necessary as our bubbles do not have much contact with objects( $F_i^{of}$ , equation 26), and we are interested in bubble movement due to the fluid. In the simulation by Kück et al. [13], $F_i^{of}$ was necessary because the bubbles were sliding down a plane.

We used a stronger $K_r$ under the surface so that a weaker $K_r$ could be used at the surface to allow the bubbles to overlap significantly creating foams. It adds greater control by having separate parameterization of repulsion forces in order to get the desired behavior in each region.

A bubble is in the below surface region when the $\phi$ value 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 -.1 which seemed to work well. If the bubble's negative radius were used as the threshold value of $\phi$, 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 10. Using a smaller portion of that bubble's radius prevents it from being considered at the surface when it is in contact with a wall. Repelling forces with the wall should prevent it from reaching

the threshold when below the surface. When a bubble has a negative value of $\phi$ that is greater than this threshold, it is in the adjacent below region.

### 5.3.2 Bubble Forces Adjacent Below the Surface

When a bubble reaches the surface, there is an attraction force between bubbles. Also, the friction between bubbles is no longer negligible.

Viscosity with other bubbles causes a bubble velocity to be dependent on the previous velocities of its neighboring bubbles (equation 32). This dependency on previous states is undesirable with the massless assumption. It would be possible to solve the equation implicitly to avoid dependency on previous velocities, but this would be costly and difficult to implement. Instead, we use the current velocities of the fluid at the position of the adjacent bubbles instead of the velocity of the adjacent bubbles themselves from the previous timestep. The new equation for the bubble friction force is

$$F_i^v = K_v(\overline{v_i^{fluid}} - v_i) \tag{37}$$

$$= K_v\left(\left[\sum_{j \in NB_i} \frac{v_j^{fluid}}{|NB_i|}\right] - v_i\right), \tag{38}$$

where $v_i^{fluid}$ is the average of the fluid velocities at bubbles intersecting bubble i, and $v_i^{fluid}$ is the fluid velocity at bubble i.

Using the fluid velocity can be justified because at the surface, it can be assumed that a bubble's velocity is most influenced by friction with the fluid at that point. The pressure just pushes the bubble up to the surface, and gravity pulls the fluid down to the surface.

Now we combine the appropriate forces (equations 29,32,33, and 37), we set the total force to zero, and we solve for velocity. This gives

$$v_i = \frac{K_{visc} v_{fluid} + K_v \overline{v_i^{fluid}} + F_i^{ra} + F^p}{K_{visc} + K_v}. \tag{39}$$

### 5.3.3 Bubble Forces Adjacent Above the Surface

If the value of $\phi$ at a bubble's position is between zero and the bubble's radius, then that bubble is in the adjacent above the surface region. It is the same as the adjacent below region

except that it is influenced by gravity instead of pressure. Substitution $F^g$ for $F^p$ into equation 39 gives

$$v_i = \frac{K_{visc} v_{fluid} + K_v \overline{v_i^{fluid}} + F_i^{ra} + F^g}{K_{visc} + K_v}. \tag{40}$$

### 5.3.4 Bubble Forces Above the Surface

Once the value of $\phi$ at a bubble position is greater than that bubble's radius, it is no longer in contact with the fluid. It can then be simulated exactly as discussed in Kück et al.[13]). It should be noted that depending on the desired effect, these bubbles may be popped because strictly water bubbles do not exist away from the water's surface.

## 5.4 SIMULATION SCHEME

The level set and particle surface is updated on a sub-cycle of the timestep used to advance the Navier-Stokes equations. 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 bubbles are regulated by the CFL condition

$$\Delta t < \frac{r_{\min}}{v_{\max}}, \tag{41}$$

where $r_{\min}$ is the smallest bubble radius, $v_{\max}$ is the largest bubble velocity, and $\Delta t$ is the timestep. This guarantees that bubbles cannot miss contact with each other. The bubble simulation step comes after the re-initialization 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 $\phi$ at any point in the bubble subcycle by interpolating between previous and past values of $\phi$. 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. The transition of a bubble between different regions is discrete.

Specifically for simulating our bubbles we used an Euler timestep 100 times smaller than the CFL condition stated in equation 40. This is arbitrary, and may be excessive, but since the

simulation of the bubbles is much faster than the fluid simulation, it did not significantly increase run times.

# 6 BUBBLE RENDERING

## 6.1 BUBBLE RENDERING FROM KÜCK ET AL. [13]

Kück et al. [13] discusses creating a contiguous foam structure from an arbitrary configuration of spherical bubbles as shown in Figure 11. This subsection exclusively discusses those methods.

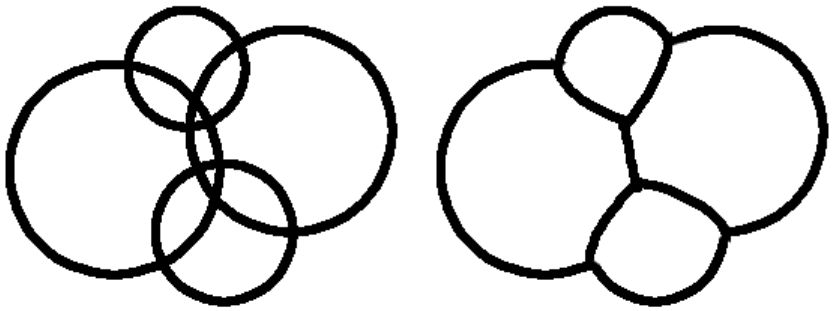

**Figure 11:** *Making spheres appear to be foam. Sphere representation (left) and corresponding foam structure (right).*

The foam is ray traced and calculations are made at each intersectios of a ray with the spheres. With time saving approximations, Kück et al. succeeded at rendering foams on the order of several thousand bubbles with these methods in reasonable time from a medium distance.

### 6.1.1 Approximated Fresnel Term for Shading Bubble Films

We use an approximation for Fresnel reflections to shade the bubble's film that is found in **[11]**[13]. The Fresnel term is the ratio of reflected to refracted non-polarized light from a dielectric (non-conducting surface) [8]. This approximation is

$$\begin{aligned} F_{approx} &= 1-\cos(\theta_i) \\ &= 1-\overline{N}\bullet\overline{I} \end{aligned}, \tag{42}$$

where $F_{approx}$ is the approximated Fresnel term, $\theta_i$ is the incident angle, $\overline{N}$ is the normalized normal, and $\overline{I}$ is the normalized incident vector. As shown in Figure 12 this creates a smooth transition from totally reflective to totally refractive.

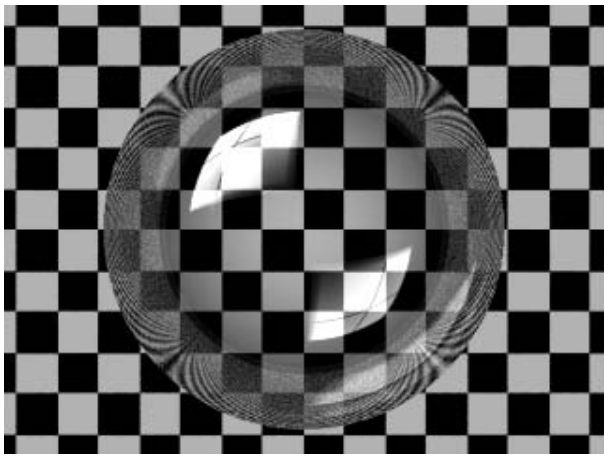

**Figure 12:** *Bubble shader from approximated Fresnel term.*

Notice that 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[13].

### 6.1.2 Two Bubbles Overlapping

The goal is to create the appearance of a contiguous foam structure from overlapping spheres. When a ray exits from two overlapping spheres, as in Figure 13, the film between the two bubbles is calculated[13]. The approximated film's position and normal is taken as the average of the two hit points on the actual spheres.

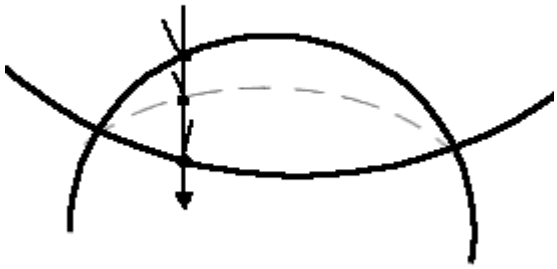

**Figure 13:** *Approximation of intersection with separating film.*

No shading calculations are done at the actual surfaces of the spheres where two spheres 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 does not apply for all viewing angles, but this approximation is inexpensive and works for most viewing angles. As shown in the figure, these methods are effective at creating the appearance of two bubbles joined together.

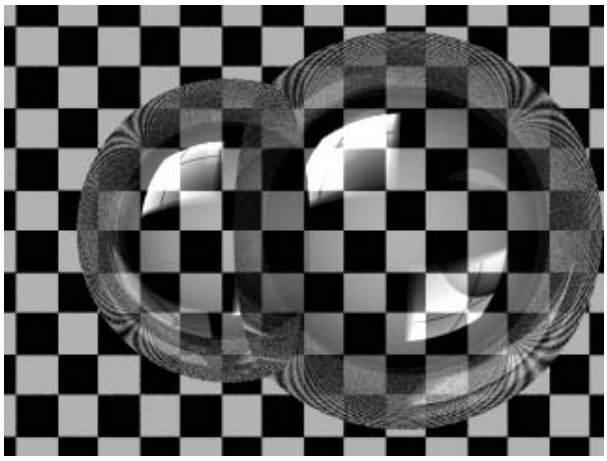

**Figure 14:** *Two-bubble cluster.*

### 6.1.3 Three Bubbles Overlapping

Kück et al [13] defined a plateau border to be where three bubbles overlap. In reality, this is where three films meet, and light is heavily scattered [13]. The space where three spheres

overlap may be much larger than an actual plateau border could be. The plateau border may be shaded with an ambient term to represent all scattered light as well as a term that represents refracted light on the plateau border [13]. In this method, the ray stops once it reaches the plateau border. In Kück et al. they are interested in dense foams that heavily scatter light and not interested in smaller bubble clusters. The use of an ambient term for a small bubble cluster is undesirable, as seen in Figure 15, because it is noticeable that light should travel beyond the region where the three spheres overlap.

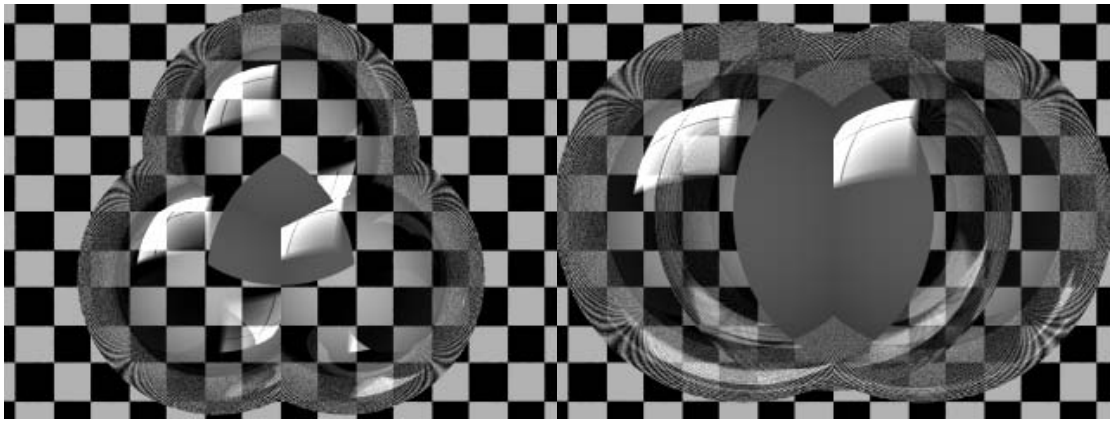

**Figure 15:** *Three-bubble cluster rendered with ambient term where three bubbles overlap.*

## 6.2 RENDERING BUBBLES WITH FLUID

Our simulation of foams differs from theirs because they simulated thousands of bubbles and we are only interested in hundreds viewed somewhat closer than in the Kück simulations. Also, the bubbles in our simulation are sloshed around vigorously and we need to be able to shade small groupings of bubbles as well as large.

### 6.2.1 Two Bubble Case and Transparent Objects

For the integration of approximated films between two bubbles and transparent objects, a special case needs to be addressed. For the case shown Figure 16, the approximated film is inside of an object, and thus it is not rendered.

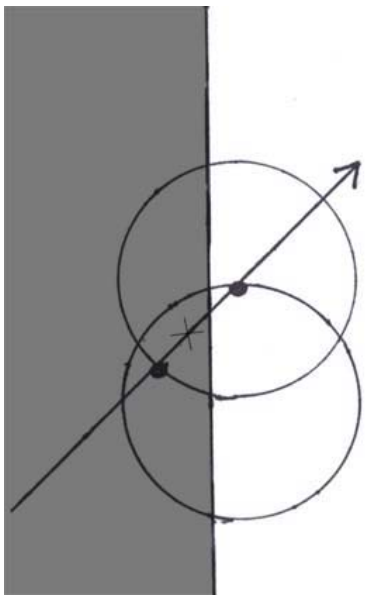

**Figure 16:** *Two-bubble case inside of object. For two bubbles overlapping, it must be checked whether the approximated film is behind the surface of an object.*

### 6.2.2 Our Treatment of Three Bubbles Overlapping

For a large number of bubbles, the Kück et al.[13] method looks okay. The problem then would be when large groups of bubbles are pushed together, and the ambient lighting suddenly appears over a large area.

For our needs it would be better to avoid an ambient term, so we decided to do something different when three bubbles overlap. It would be acceptable to use more expensive rendering methods to render our bubbles because we are rendering fewer bubbles and speed is not as high of a priority.

It would be nice to continue the approximated films from 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 accomplish that task.

Instead we treated the three bubble overlapping case similarly to the two bubble overlapping case. As seen in Figure 17, once a ray enters an area where the bubbles overlap, shading is not 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 bubble two case, the

average point is inside of the two bubbles so we can send the reflection ray from that point).

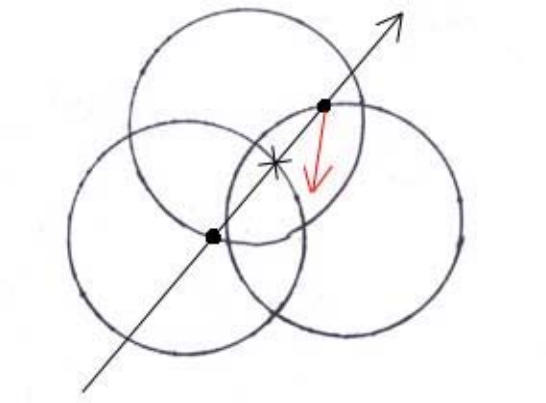

**Figure 17:** *Modified method for shading where three bubbles overlap. Reflection ray (red) is sent from the final hit point because it is unknown which bubbles the averaged point (x) is inside.*

Due to an implementation error, 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 17). After fixing this "bug", the rendered images looked less desirable, so it was changed back so that approximated films always use the original hit point (until the ray travels past the three bubble region and there is a new "original" hit point).

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. As seen in Figure 18 it looks better than using the ambient term of Figure 15.

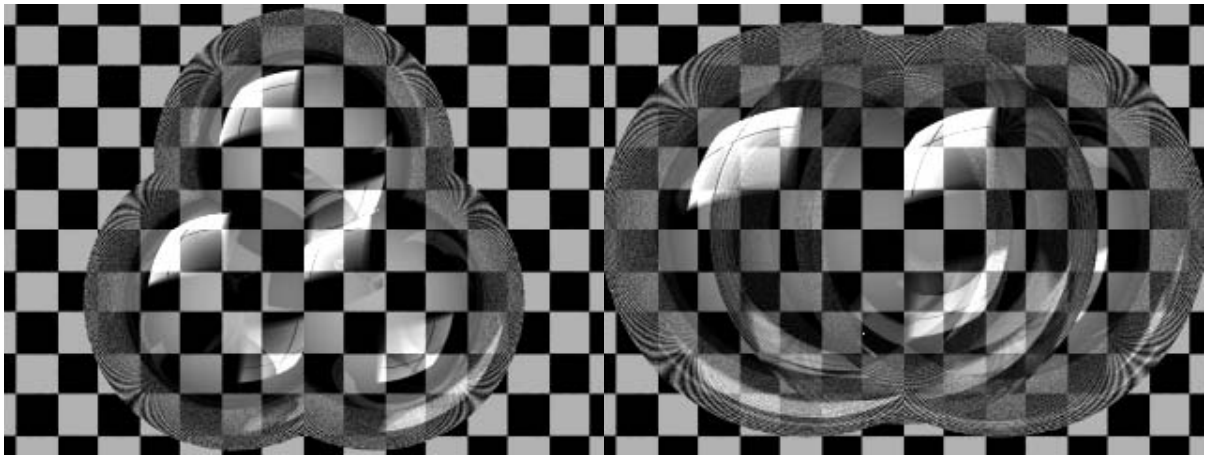

**Figure 18:** *Three-bubble cluster rendered with our modified method.*

The hope is that by rendering the bubbles with this method, these small regions where three bubbles overlap will not be noticeable when seen from a distance.

### 6.2.3 True Fresnel Term for Fluid Surfaces

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 the actual Fresnel term for calculating the reflection ratio, versus the approximated Fresnel term for bubble films. The Fresnel equation [8] can be expressed as

$$F_\lambda = \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left(1+\frac{\left[c(g+c)-1\right]^2}{\left[g(g-c)+1\right]^2}\right), \tag{43}$$

where $c = \cos(\theta_i) = \overline{N} \bullet \overline{I}, g^2 = \eta_\lambda^2 + c^2 - 1$ and $\eta_\lambda = \eta_{t\lambda} / \eta_{i\lambda}$. $\eta_{t\lambda}$ is the index of refraction for the object hit, and $\eta_{i\lambda}$ is the index of refraction for the object that the ray is already inside of.

As seen in Figure 19, light reflects differently whether hitting a fluid surface from inside or outside, or hitting a bubble film. It should be noted that after calculating the Fresnel term, specular highlights are rendered as the actual specular reflection of lights (which is what a specular highlight actually is). 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.

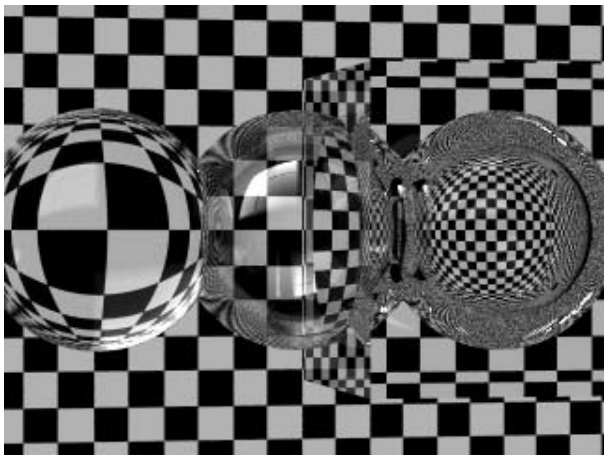

**Figure 19:** *Refracting water surface. From left to right: water sphere, half submerged bubble showing simulated Fresnel reflections on left half, and totally submerged bubble.*

### 6.2.4 Fluid and Obstacle Boundaries

While a level set is very powerful at defining numerous shapes, there is little possibility that a level set could be defined so that the zero level set sits flush up against an object in the simulation (unless that object is a plane). If the level set is not allowed to overlap, 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 air and another between the air and the object. The raytracer should only render one surface, the surface between the fluid and the object. For this case, the raytracer would have to check ahead, and if the two hits happen close enough together, it would render them as one hit. 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.

### 6.2.5 Hierarchy Between Overlapping Objects

Now the renderer only has to worry about rendering the boundaries of the object contained in the fluid. It does not render 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 the 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 19 and Figure 20.

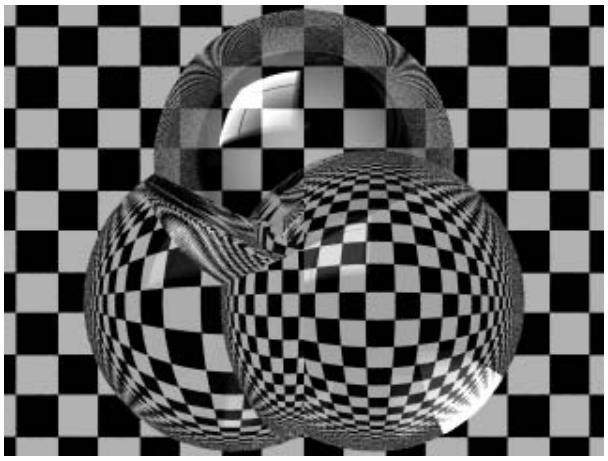

**Figure 20:** *Hierarchy of surfaces. Water(left), glass(right), and bubble(top).*

It should be noted that 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).

# 7 RESULTS AND SUMMARY

## 7.1 ISSUES

Several assumptions and approximations used in this work lead to limitations. We discuss these below.

### 7.1.1 Spherical Bubbles

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 larger bubbles. As in Figure 21, a large spherical bubble looks strange at the surface. A similar problem occurs when viewing the bubbles up close, and these methods work best for rendering bubbles at a medium distance.

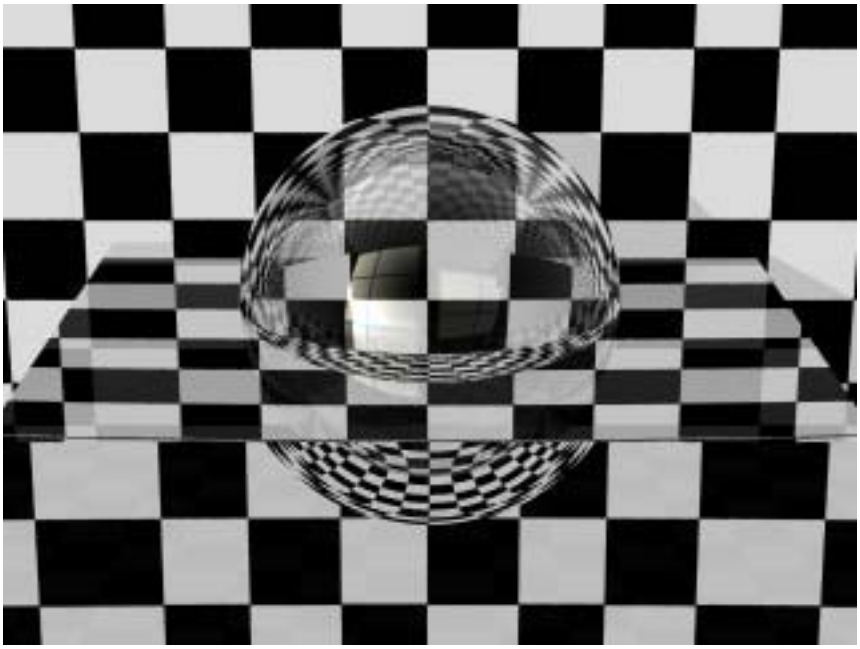

**Figure 21:** *Large spherical bubble at fluid surface.*

### 7.1.2 Unfeasible bubbles

Marker particles may escape occasionally when it is unrealistic and undesired for a bubble to form. While we take steps against 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 animator has the desired water motion, the animator could move on to animating the bubbles. Bubbles created from marker particles could be used as the basis for realistic bubbles created from mixing fluid. An animator could combine our techniques with standard particle tools to create bubble motion as desired.

### 7.1.3 Air Pockets Do Not Disappear Immediately

Just making the sign of $\phi$ 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 $\phi$ (which 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 $\phi$ 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 $\phi$ 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 exist for more than a few frames. As seen in Figure 22, 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.

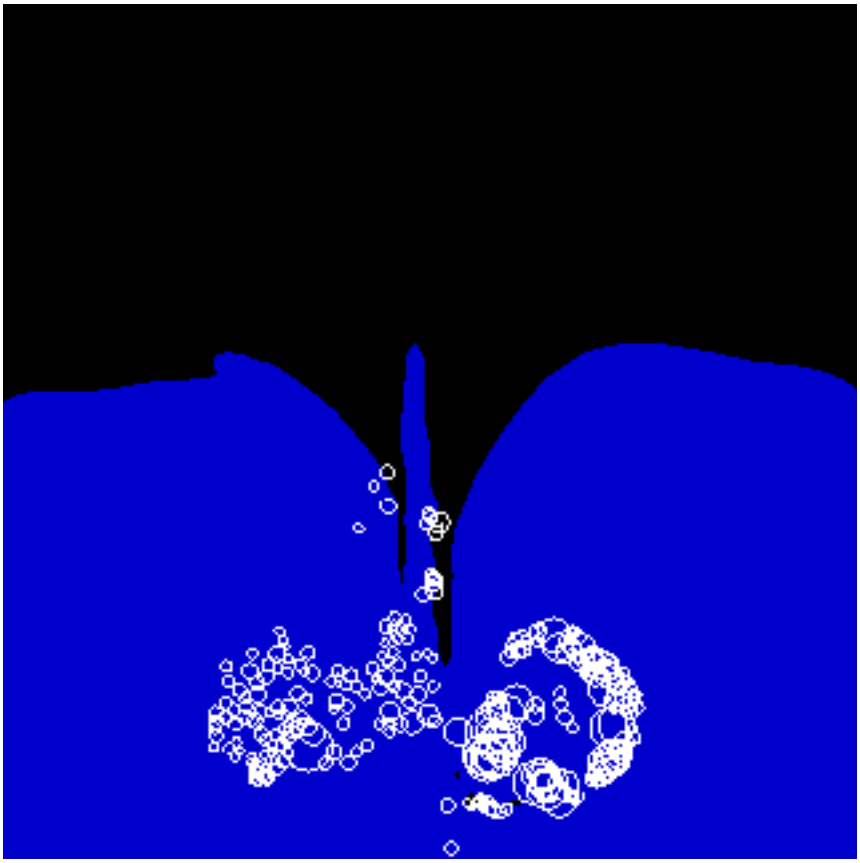

**Figure 22:** *Small residual pieces of air pocket may persist for a few frames.*

## 7.2 BUBBLES CREATING MORE REALISTIC FLUID

Our methods of adding bubbles to fluid simulation and rendering can enhance the overall realism of creating the appearance of a splashing fluid for computer graphics. In reality, bubbles form in a splashing liquid. The inclusion of bubbles may be more convincing and aesthetically pleasing than not having bubbles. 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. This is a definite case in which bubbles would form in reality. Our methods provide an inexpensive way of dealing with air pockets without simulating the air.

As seen in Figure 23 and Figure 24, air pockets are removed. The air pocket disappears more abruptly than it would if the air pocket were not removed and the fluid simulation were allowed to engulf the air pocket; however, the effect is the same, as the disappearance of air into the fluid looks unrealistic.

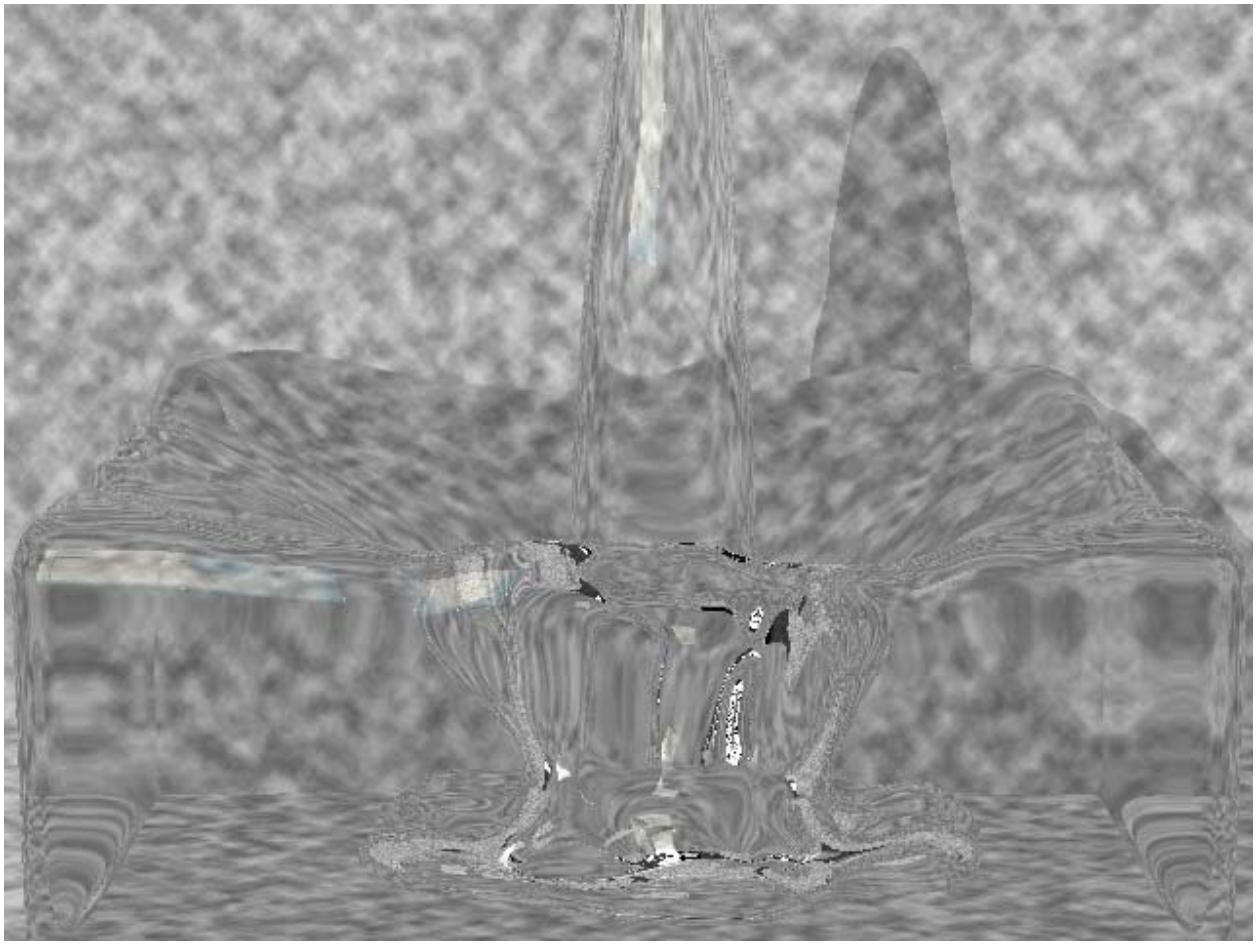

**Figure 23:** *Frame of animation before removal of air pocket.*

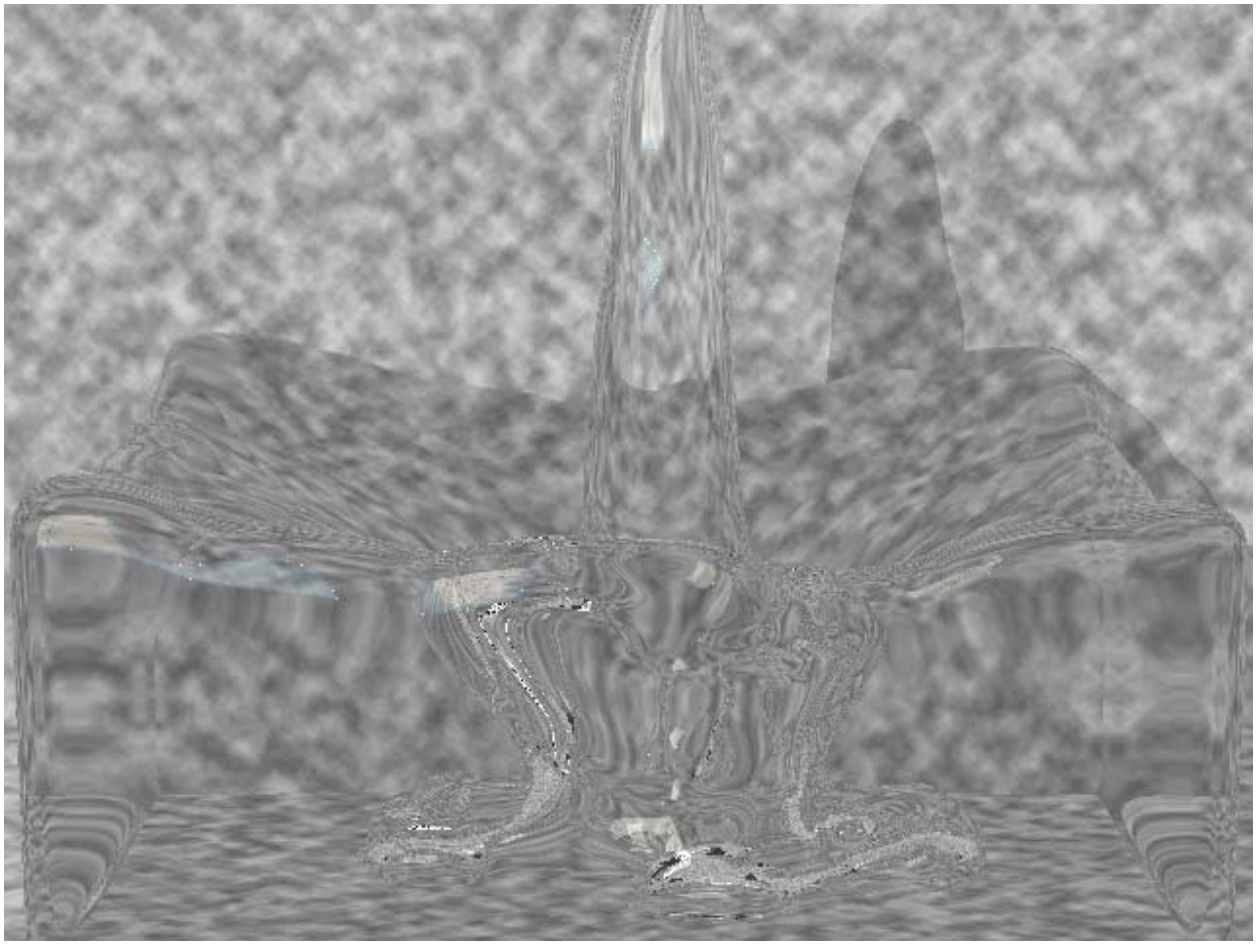

**Figure 24:** *Frame of animation after removal of air pocket.*

The creation of bubbles hides the disintegration of air pockets and creates more true to life animations as seen in Figure 25.

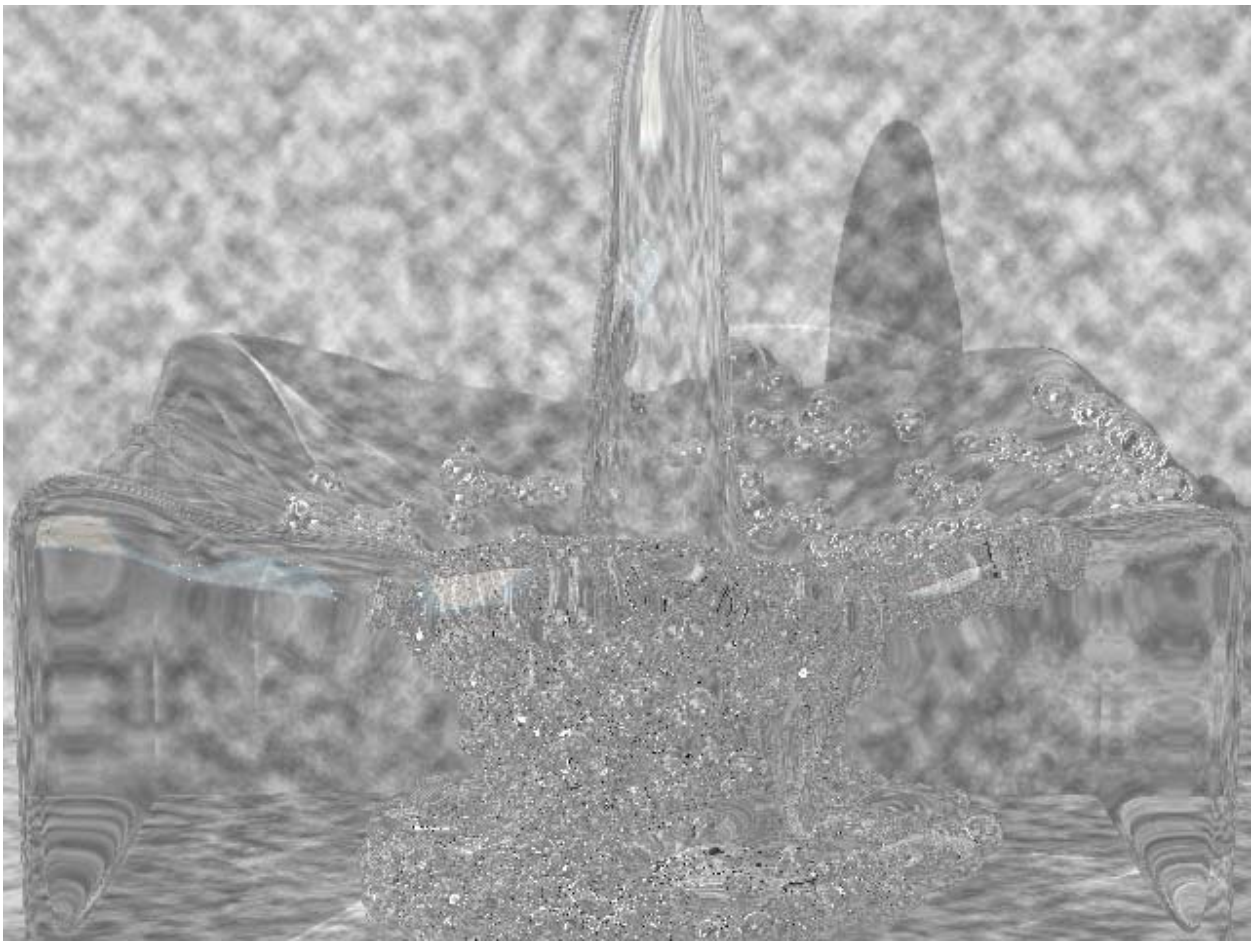

**Figure 25:** *Frame of animation with bubbles instead of disappearing air pocket.*

Also, by creating bubbles, the added detail of the bubbles may distract from other problems with the fluid's appearance. Overall, these methods are a good way of creating realistic bubbles if viewed from a medium distance as seen in Figure 26 and Figure 27.

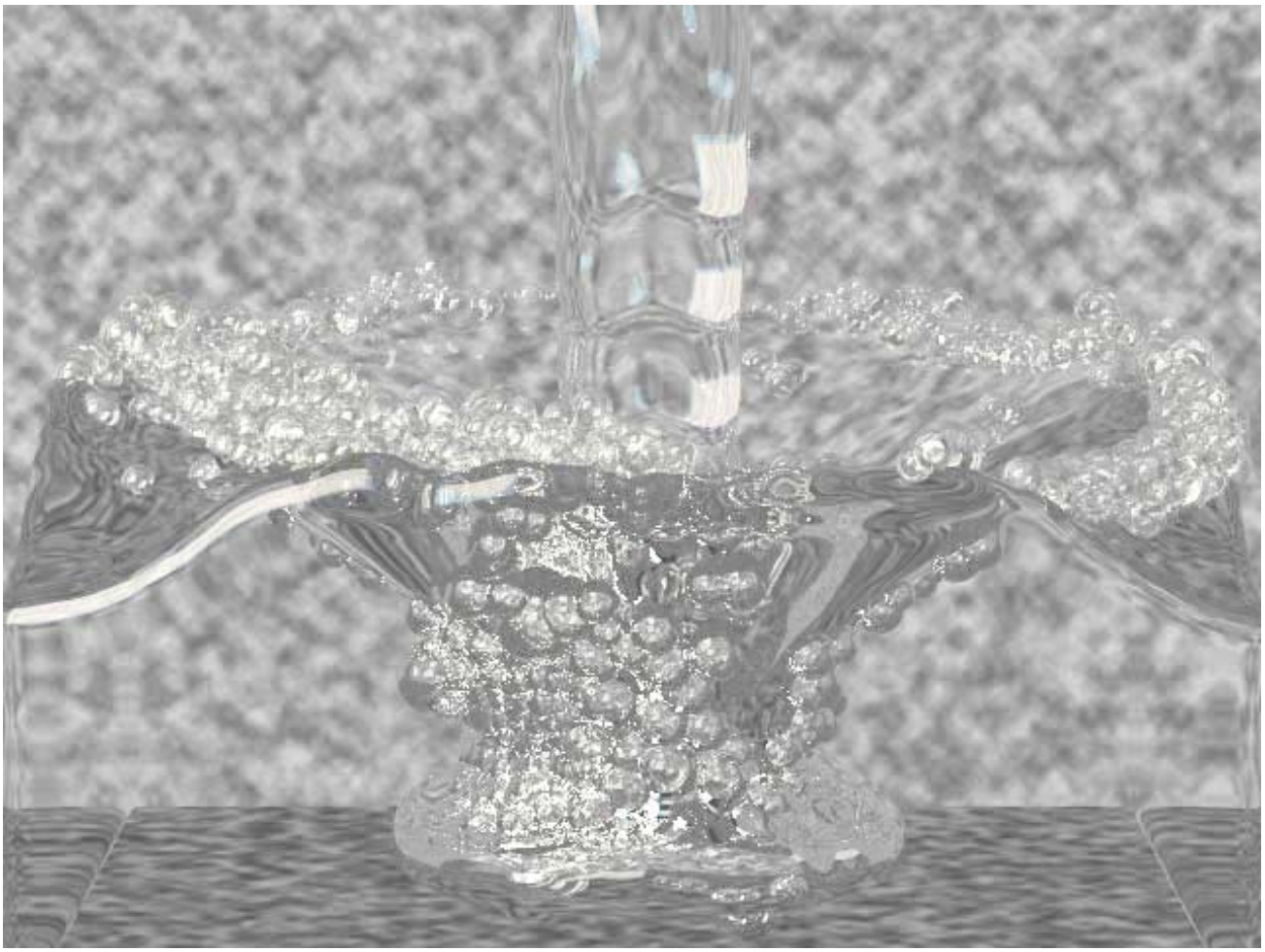

**Figure 26:** *High quality frame of fluid column splashing in the center.*

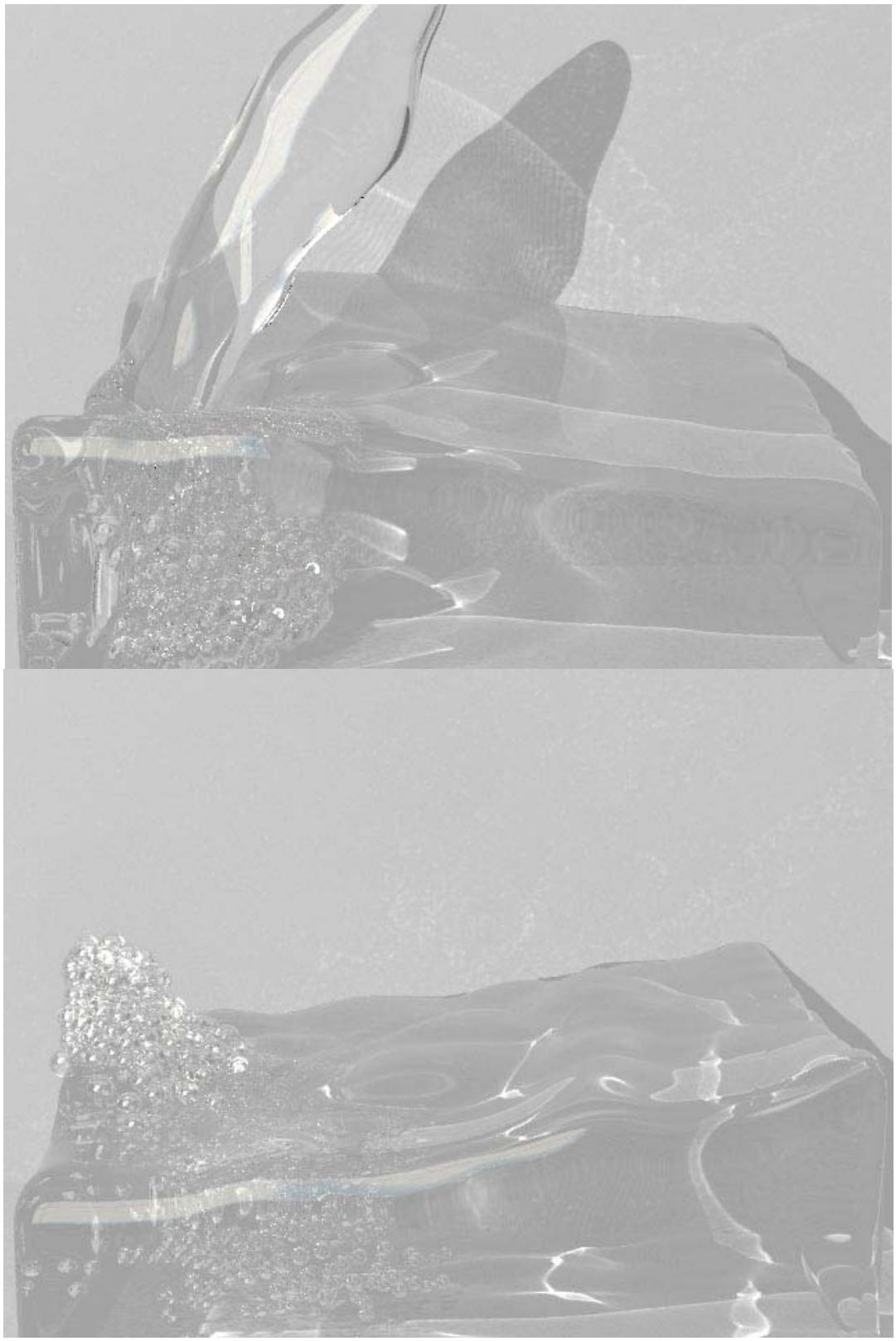

**Figure 27:** *Frames from animation with splash on the side.*

# REFERENCES

[1] D. Adalsteinsson and J. Sethian, "The Fast Construction of Extension Velocities in Level Set Methods," *J. of Comput. Phys*., vol. 148, pp. 2-22, 1999.

[2] F. Bolton, "A Computer Program for the Simulation of Two-Dimensional Foam", www.tcd.ie/Physics/Foams/plat.html, 1990.

[3] S. Chen, D. Johnson, and P. Raad, "Velocity Boundary Conditions for the Simulation of Free Surface Fluid Flow", *J. of Comput. Phys*., vol. 25, pp. 749-778, 1995.

[4] D. Durian, "Foam Mechanics at the Bubble Scale", *Physical Review Letters,* vol. 75, pp. 4780–4783, 1995.

[5] D. Durian, "Bubble-Scale Model of Foam Mechanics: Melting, Nonlinear Behaviour and Avalanches", *Physical Review E*, vol. 55, pp.1739–1751, 1997.

[6] D. Enright, S. Marschner, and R. Fedkiw, "Animation and Rendering of Complex Water Surfaces*", ACM Trans. on Graphics (Proc. SIGGRAPH '02)*, vol. 21, no. 3, pp. 736-744, July 2002.

[7] R. Fedkiw, T. Aslam, B. Merriman, S. and Osher, " A Non-Oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method)", *J. of Comput. Phys*., vol. 152, pp. 457-492, 1999.

[8] J. Foley, A. Van Dam, S. Feiner, and J. Hughes, *Computer Graphics Principles and Practice*. Reading, Mass.: Addison-Wesley, 1990.

[9] N. Foster and R. Fedkiw, "Practical Animation of liquids", *Proc. SIGGRAPH '01*, 23-30, 2001.

[10] N. Foster and D. Metaxes, "Realistic Animation of liquids", *Graphical Models and Image Processing*, vol. 58, pp. 471-483, 1996.

[11] A. Glassner, "Andrew Glassner's Notebook: Soap Bubbles, Part 2", *IEEE Computer Graphics and Applications*, vol. 20, no.6, 99-109, 2000.

[12] F. Harlow and J. Welch, "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with a Free Surface", *The Physics of Fluids*, vol. 8, pp. 2182-2189, 1965.

[13] H. Kück, C. Vogelgsang, and G. Greiner, "Simulation and Rendering of Liquid Foams", *Proc. Graphics Interface '02*, pp. 81-88, 2002.

[14] J. Stam, "Stable Fluids", *proc. SIGGRAPH '99*, 121-128, 1999.

[15] D. Weire and S. Hutzler, *The Physics of Foams*. Oxford, UK: Oxford University Press, 1999.

# VITA

Shannon Thomas Greenwood
4114 Antelope Trail
Temple, TX 76504
(281) 221-7594
shannon_greenwood@hotmail.com

**Education**

Texas A&M University — College Station, TX
M.S. in Visualization Science, May 2004

Texas A&M University — College Station, TX
B.S. Computer Engineering CS track, August 2001
GPA 3.96/4.0, Summa Cum Laude, University Honors

**Training & Software**

C, C++, OpenGL, MEL
Windows/ DOS, Unix, Linux
Alias/Wavefront Maya, Adobe Photoshop/After Effects

**Experience**

National Instruments, Austin, TX
*Engineering Coop*

*3rd Term - Application Engineering* — Summer 2000
*Provided technical support directly to customers over telephone
*Designed and Implemented web remote control demo

*2nd Term - Research and Development* — Fall 1999
*Created a proof of concept of a PDA acquiring measurement readings from a monitorless PC using IrDA protocol

*1st Term - Research and Development* — Spring 1999
*Exposed to kernel level driver development
*Created debug macro for manipulating debug prints while running SoftIce Debugger