Coupling Water and Smoke to Thin Deformable and Rigid Shells

Eran Guendelman[*] Stanford University Industrial Light + Magic
Andrew Selle[*] Stanford University Intel Corporation
Frank Losasso[*] Stanford University Industrial Light + Magic
Ronald Fedkiw[†] Stanford University Industrial Light + Magic

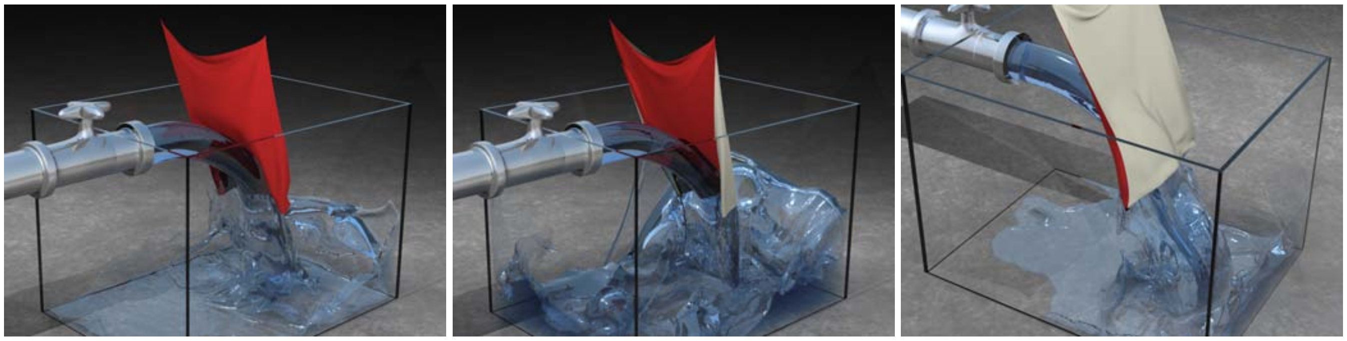
Figure 1: Water and cloth interacting with full two way coupling ($256 \times 256 \times 192$ effective resolution octree, 30K triangles in the cloth).

Abstract

We present a novel method for solid/fluid coupling that can treat infinitesimally thin solids modeled by a lower dimensional triangulated surface. Since classical solid/fluid coupling algorithms rasterize the solid body onto the fluid grid, an entirely new approach is required to treat thin objects that do not contain an interior region. Robust ray casting is used to augment a number of interpolation, finite difference and rendering techniques so that fluid does not leak through the triangulated surface. Moreover, we propose a technique for properly enforcing incompressibility so that fluid does not incorrectly compress (and appear to lose mass) near the triangulated surface. This allows for the robust interaction of cloth and shells with thin sheets of water. The proposed method works for both rigid body shells and for deformable manifolds such as cloth, and we present a two way coupling technique that allows the fluid's pressure to affect the solid. Examples illustrate that our method performs well, especially in the difficult case of water and cloth where it produces visually rich interactions between the particle level set method for treating the water/air interface and our newly proposed method for treating the solid/fluid interface. We have implemented the method on both uniform and adaptive octree grids.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: water, smoke, cloth, shells, rigid bodies

1 Introduction

Water and smoke both possess a large number of degrees of freedom and thus produce visually rich motion especially when interacting with solid objects. This makes these media both interesting and popular from a storytelling or entertainment perspective. It can be rather costly, uncomfortable or impossible to capture the desired fluid interaction with actors on film, and thus simulation techniques for these phenomena have become popular in recent years. Often, two way coupling between fluids and solids is not desirable, since the animator does not want the fluid changing or resisting their artistic development. On the other hand, animators have difficulty when the solid has many degrees of freedom, e.g. cloth, and have resorted to simulation to obtain the desired effects. Moreover, thin and light weight objects need to feel the effect of the fluid, especially heavy fluids such as water, in order to make their animation plausible. Thus, the two way interaction of thin deformable high degree of freedom solids and heavy liquids with interfaces is highly desirable.

There are various computational methods for simulating fluids, solids and their coupling. Typically fluids such as water (e.g. [Foster and Fedkiw 2001; Enright et al. 2002]) are simulated using Eulerian numerical methods with a fixed mesh that material moves through, whereas solids such as cloth (e.g. [Bridson et al. 2002; Choi and Ko 2002; Baraff et al. 2003]) are simulated with a Lagrangian numerical method where the mesh moves with the material. At least as far back as [Noh 1964] (see also [Benson 1992] for a review), two way coupling has been carried out with the fluid's pressure providing forces to the solid, and the solid's velocity providing boundary conditions for the fluid. Although this often used method works well for problems where the solid object is thick enough to be resolved by the fluid's grid, see e.g. [Yngve et al. 2000], it suffers from significant aliasing errors for thin objects. This is exacerbated by the infinitesimally thin objects made up of triangles that we consider here, and recent methods (e.g. [Carlson et al. 2004]) cannot be applied to these types of problems.

Very little research has been carried out on algorithms that couple infinitesimally thin Lagrangian-based solids to Eulerian-based fluids, and few computational strategies exist. Moreover, they are mostly focused on single phase fluids, whereas our main interest is fluids with interfaces such as between water and air. Probably the most common strategy for *single phase* fluids is based on the immersed boundary method of [Peskin 1972; Peskin 2002], and [Zhu and Peskin 2002] used this method to calculate the motion of a thin flexible filament (a curve) in two spatial dimensions. A thin solid object feels and reacts to fluid forces as molecules collide against it, and the net force on the thin solid comes directly from the pres-

[*]e-mail: {erang, aselle, losasso}@stanford.edu
[†]e-mail: fedkiw@cs.stanford.edu

sure differential across it. The immersed boundary method cannot handle this pressure jump and instead forces the pressure to be continuous across the thin solid, and thus (nonexistent) pressure jumps cannot be used to apply forces to the solid. Instead, they simply set the solid velocity to be equal to the velocity of the surrounding fluid, and use ad hoc methods to provide resistance to the fluid motion. For example, [Zhu and Peskin 2002] smeared out the filament over a number of grid cells converting it into a higher density fluid, and added artificial forces to the right hand side of the Navier-Stokes equations. Similar to penalty methods for rigid body contact constraints, these forces can only coerce a desired fluid reaction and often require small time steps for stability and accuracy.

[Kang et al. 2000] pointed out that smeared out pressures profiles (such as those used in the immersed boundary method) can cause parasitic currents when used to make the velocity divergence free (see also [Génevaux et al. 2003]). A key to our method is the replacement of penalty forces with analytic constraints on the fluid velocity forcing it to flow as dictated by the velocity of the solid. Heuristically similar to the analytic methods of [Baraff 1993] for solving contact phenomena in rigid bodies, we replace the stiff inaccurate penalty forces of the immersed boundary method with a robust constraint that requires the solution of a linear system of equations greatly reducing the errors. Conveniently, we are already solving a linear system for the pressure, and it is readily modified to include the no flow constraint exactly as opposed to the only approximate enforcement via penalty forces. This is essentially a sharp interface approach similar in spirit to the immersed interface method (see e.g. [Li and Lai 2001]). However, we note that neither the immersed interface method nor the immersed boundary method has been used to solve solid/fluid coupling problems in the presence of liquid interfaces or thin films as we do here.

2 Previous Work

Simulation of the incompressible three dimensional Navier-Stokes equations was popularized by [Foster and Metaxas 1997b] and later made more efficient by [Stam 1999] via the use of semi-Lagrangian advection techniques. [Fedkiw et al. 2001] increased the small scale details with the use of a vorticity confinement term. These equations have also been augmented to model fire [Lamorlette and Foster 2002; Nguyen et al. 2002], particle explosions [Feldman et al. 2003], and even the interior of deformable objects [Nixon and Lobb 2002]. [Rasmussen et al. 2003] combined interpolation with two dimensional simulations to create three dimensional nuclear explosions, [Treuille et al. 2003; Fattal and Lischinski 2004] proposed methods for control, and [Stam 2003] solved these equations on surfaces creating beautiful imagery.

[Kass and Miller 1990] solved a linearized form of the three dimensional Navier-Stokes equations for water that interacted with a variable height terrain. [Chen and Lobo 1994] solved the two dimensional Navier-Stokes equations using the pressure to define a height field. The full three dimensional Navier-Stokes equations were solved in [Foster and Metaxas 1996; Foster and Metaxas 1997a] using a particle in cell approach. A hybrid particle and implicit surface approach to simulating water was proposed in [Foster and Fedkiw 2001], which led to the particle level set method of [Enright et al. 2002]. Additional work includes the modeling of surface tension [Enright et al. 2003; Hong and Kim 2003; Cohen and Molemaker 2004; Losasso et al. 2004], viscoelastic fluids [Goktekin et al. 2004], control [McNamara et al. 2004; Mihalef et al. 2004], and octree data structures [Losasso et al. 2004]. Level set methods for simulating liquids have enjoyed popularity in recent films including “Terminator 3” [Rasmussen et al. 2004], “Pirates of the Caribbean” [Rasmussen et al. 2004], “The Day After Tomorrow” [Iversen and Sakaguchi 2004], “The Cat in the Hat” [Cohen and Molemaker 2004] and “Scooby Doo 2” [Wiebe and Houston 2004; Houston et al. 2004].

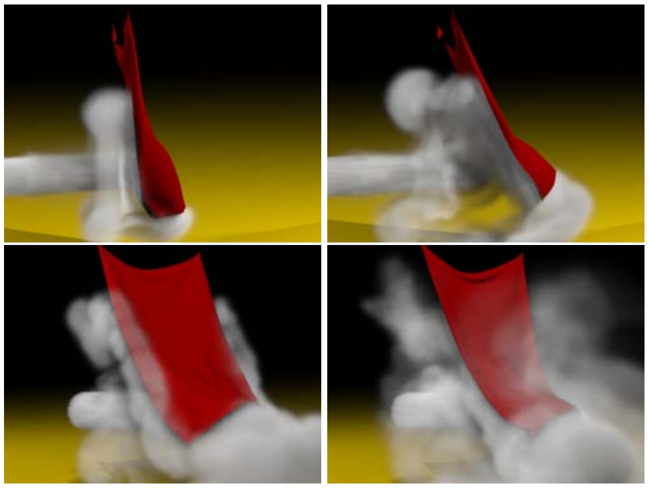
Figure 2: Two way coupled cloth and smoke ($210 \times 140 \times 140$ uniform grid, 30K triangles in the cloth).

Various authors have used simplified fluid dynamics to blow around solid objects, e.g. [Wejchert and Haumann 1991; Wei et al. 2003], and many have used simplified wind models to simulate flags flapping in the wind, e.g. [Ling et al. 1996]. [Hadap and Magnenat-Thalmann 2001] used gridless SPH techniques to couple air flows to hair simulation, but since hair is one-dimensional it does not restrict or contain the fluid as cloth does. SPH models for water were considered in [Premoze et al. 2003; Muller et al. 2003], and methods of this type were coupled to deformable solids in [Muller et al. 2004] using virtual boundary particles. In fact, [Kondoh et al. 2004] coupled an SPH model for water to thin deformable cloth pointing out that particle based fluid methods can be coupled without leaking using robust point face collisions, although their method will leak if the time step is not chosen sufficiently (sometimes severely) small. Of course, this can be alleviated with a more robust point face collision method as in [Bridson et al. 2002]. The drawback of using SPH methods is that it is difficult to obtain the smooth liquid surfaces characteristic of level set methods, and recently [Carlson et al. 2004] proposed a method for the two way coupling of rigid bodies to level set based fluid simulations. They first rasterize the rigid body velocity onto the grid, and then solve the fluid equations everywhere treating the rasterized rigid body velocities as if they were fluid (this was also done in [Foster and Fedkiw 2001] for modeling slip boundary conditions). Then they modify the velocities in the rigid body region to account for collisions and buoyancy before averaging them to a valid rigid body velocity with a constant translational and rotational component. The authors point out that their method leaks if the objects are too thin (whereas we consider arbitrarily thin objects), and deformable materials were not considered.

We provide examples of water and smoke interacting with thin rigid bodies and cloth. The novelty of the method is in the treatment of the fluid and the interaction between the fluid and the solid, not in the simulation of the solids themselves. Thus, we use the basic cloth model from [Bridson et al. 2003] including their bending formulation (see also [Grinspun et al. 2003]), and the self-collision algorithm of [Bridson et al. 2002]. We note that there are many other interesting strategies for cloth including the dynamics model proposed in [Baraff and Witkin 1998], the bending model proposed in [Choi and Ko 2002], and the self-interference untangling strategy

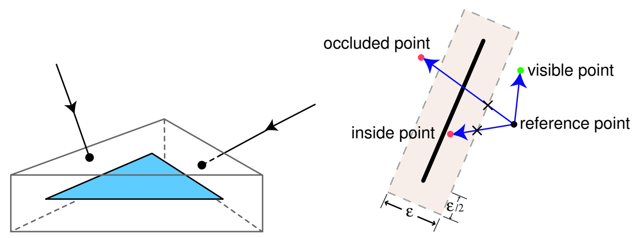

Figure 3: We intersect with a triangle wedge that is formed by extending the edges and face in normal directions by $\epsilon/2$ (left). Given a reference point, another point can be classified as *visible*, *inside* an object, or *occluded* by the result of a ray cast (right).

of [Baraff et al. 2003]. We use a basic method for rigid body simulation, which doesn't need to be any more sophisticated than that in [Hahn 1988; Moore and Wilhelms 1988] for our purposes. The interested reader is also referred to [Baraff 1994; Guendelman et al. 2003] and the references therein for contact and collision handling techniques.

3 Robust One-Sided Interpolation of Data

Our goal is to completely prevent the leaking of smoke and water across thin rigid and deformable solids represented by moving triangles. To do this, we use visibility and occlusion to determine which point combinations can be used to produce interpolations, derivatives, etc. of variables such as ϕ, ρ and $\mathbf{u}$. This is accomplished via robust ray casting against thickened triangle wedges as in [Bridson et al. 2002], see Figure 3 (left). From the perspective of any reference point in space, the world is broken up into three regions: *visible* points, *occluded* points, and points *inside* some triangle wedge of the object. This partitioning is accomplished by casting a ray from the reference point to the point in question as shown in Figure 3 (right). The triangle wedges guarantee that only visible points are labeled *visible*, but may incorrectly mislabel some visible points as being *inside* the object or even *occluded* near boundaries leading to a fuzzy interpretation of the object that is robust against roundoff errors. Given a reference point, if one rules out all *occluded* and *inside* points when constructing stencils for interpolation, differentiation, etc. at this reference point, then valid one sided approximations are guaranteed.

When a thin solid moves, a point originally on one side of the object surface may be swept over by the surface and end up on the other side of it. In this case, the values contained by that point are invalidated for all subsequent interpolation, since it represents information from the other side of the object. Detecting such points is crucial to preventing leaks and is accomplished on a per-triangle basis. Each time step, we move the triangle nodes with linear trajectories, and consider a point *invalid* if it intersects the triangle itself (at the center of the wedge) during the time step. Checking this amounts to solving a cubic equation as in [Bridson et al. 2002]. For robustness, we additionally consider a point *invalid* if it is either inside the triangle wedge at the beginning or at the end of a time step. Any point that does not start or stop *inside* a triangle wedge will robustly register a collision with an interior triangle if it crosses from one side of the object to the other. In the case of octrees, refinement leads to new point values that are also marked as *invalid*. Coarsening only involves the removal of nodes, and thus nothing special need be done.

We provide valid values for *invalid* nodes using a Gauss-Jacobi iterative scheme to propagate information. Each iteration, every *invalid* node is assigned the average of its valid *visible* one ring neighbor values and marked *valid*. This technique of averaging uncovered points is similar to the blending methods used by others, see e.g. [Benson 1992]. Complicated object geometry or folding may produce nodes that are still *invalid* after all iterations are complete. These nodes have no *valid visible* neighbors, and thus we again iterate in a Gauss-Jacobi fashion except this time using specially chosen values when visibility rays intersect an object. For example, we use the object velocity, a zero density, an ambient or object temperature, the positive distance to the object, etc.

A standard axis-aligned box hierarchy is used for the triangulated surface accelerating intersection tests, etc. Moreover, for each triangle, a slightly enlarged bounding box is used to label all the voxels from the fluid simulation that are in close proximity to the surface thus possibly requiring special treatment.

4 Fluid Simulation

As proposed in [Fedkiw et al. 2001], we ignore viscous effects and use the inviscid form of the Navier-Stokes equations,

$$\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \mathbf{f}, \quad (1)$$

$$\nabla\cdot\mathbf{u} = 0, \quad (2)$$

where $\mathbf{u} = (u,v,w)$ is the velocity field, $\mathbf{f}$ accounts for the external forces, and the density of the mixture has been absorbed into the pressure p. Although we have implemented the algorithm on both uniform and octree grids, the exposition is primarily geared towards uniform grids with octree grids discussed only when the extension is not obvious. Otherwise, we refer the reader to [Losasso et al. 2004] for more details. In addition, *we have devised a novel node based fluid solver* with the velocity, temperature, density and level set values all defined on the nodes of both the uniform and octree grid. See section 4.2.

4.1 Computing the Intermediate Velocity

We trace a semi-Lagrangian ray from $\mathbf{x}$ to $\mathbf{x} - \Delta t\mathbf{u}$, for each nodal velocity, intersecting it with a triangle wedge that is double the usual size (i.e. using $\epsilon' = 2\epsilon$ in Figure 3 (left)). This guarantees that the base interpolation point is *visible* to the point we are updating, and that the subsequent 8 rays we send out from this base interpolation point can accurately predict visibility (through a 2 ray path) for the interpolation stencil. If any of these 8 secondary rays intersect the object, we use the local object velocity for that term in the trilinear interpolation formula. After computing the intermediate velocity field, the object is moved to its new location and we label all nodes colliding with the moving object as *invalid*. Finally, these nodes are given *valid* values as described in section 3.

For water, gravity is simply added to each nodal velocity in the usual manner. For smoke, we add source terms that depend on the smoke's density ρ and the fluid's temperature T, i.e. $\mathbf{f} = -\alpha\rho\mathbf{z} + \beta(T - T_a)\mathbf{z}$ where $\mathbf{z}$ is the upward direction, α and β are tunable parameters, and T_a is the ambient temperature. The smoke's density and the fluid's temperature are treated (advection, visibility, cross over, invalid, etc.) together with the velocity as explained above, using a zero density and T_a for visibility rays that intersect the object.

To compute the vorticity confinement force at each grid node, we calculate the curl of the velocity field $\omega = \nabla\times\mathbf{u}$ using the six adjacent nodal velocity vectors with each replaced by the object velocity if it is not *visible*. Then we compute the vorticity magnitude. Its gradients, $\nabla|\omega|$, are computed using central differencing

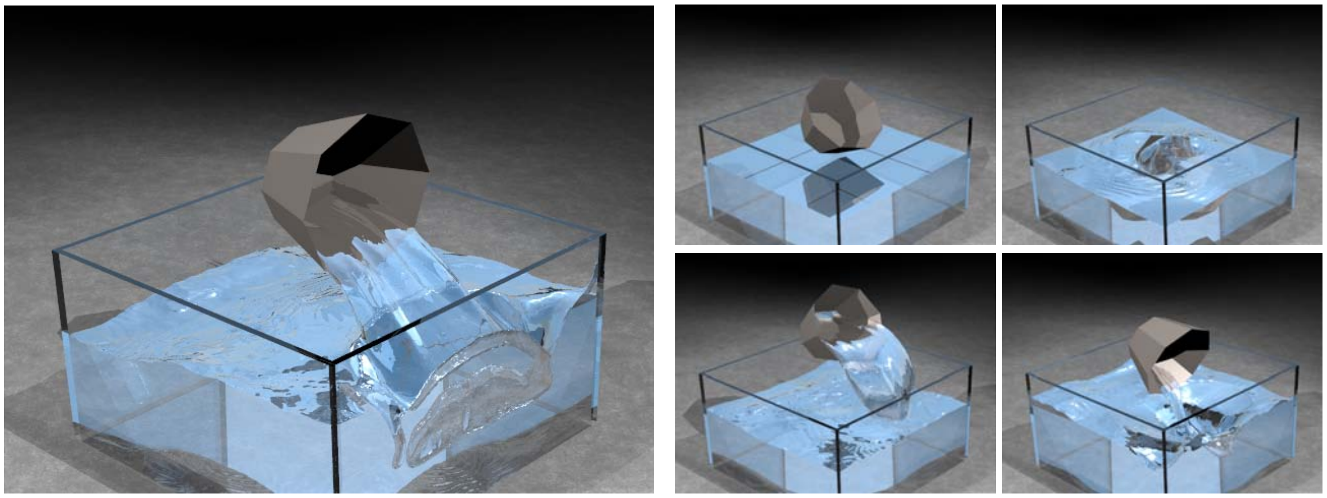
Figure 4: A thin rigid kinematically controlled cup is filled with water, and then poured out ($160 \times 192 \times 160$ effective resolution octree).

of the six neighboring values of the vorticity with each replaced by the vorticity of the center node if it is not *visible*. These gradients are normalized to obtain the vorticity location vectors $\mathbf{N}$ which are then used to compute the source term due to vorticity at the node, $\mathbf{f} = \hat{\varepsilon}\Delta x(\mathbf{N} \times \omega)$.

4.2 Node Based Fluid Solver

We solve for the pressure on the standard MAC grid. This could be done by averaging the nodal velocities to the faces, projecting the face velocities to be divergence free, and then averaging the result back to the nodes. However, these extra averaging steps deteriorated the quality of our results so we designed an alternate method. We start by defining the initial velocity field on the faces instead of the nodes. Then at each step, we first average the face values to the nodes before computing the intermediate velocity $\mathbf{u}^*$ on the nodes as described above. Then, instead of averaging this intermediate velocity back to the faces, we compute a scaled force on each node as $\triangle\mathbf{u} = \mathbf{u}^* - \mathbf{u}^n$ and average this scaled force back to the faces instead. The scaled force is then used to increment the persistent face velocities to obtain the intermediate velocity on the faces. This alternative method smears out the incremental forces instead of the persistent velocities, and thus leads to excellent results competitive with the standard MAC scheme. To understand the difference, consider that averaging velocities back and forth causes dissipation independent of $\triangle t$ and even leads to dissipation when $\triangle t = 0$. On the other hand, the smearing of our forces is scaled by $\triangle t$ and vanishes as $\triangle t$ approaches 0. Overall, this method allows us to combine a simple and efficient node based scheme for computing the intermediate velocity with a robust face based scheme for making the intermediate velocity divergence free.

This mixed node/face based scheme can be used to accelerate and simplify any Navier-Stokes solver, but a few modifications are required in our case. First, when averaging from the faces to the nodes, rays are traced to the appropriate four faces and all *visible* faces are used to compute an average velocity on the node. If no faces are *visible*, we average the local object velocity from the locations where the visibility rays intersected the object. Then when averaging the scaled forces back to the faces, the appropriate rays are traced and all *visible* nodes are used to compute the average scaled force on a face. If no nodes are *visible*, we replace the face velocity with the average obtained using the object velocities determined by the visibility rays, i.e. scaled forces are not used in this case.

4.3 Solving for the Pressure

The intermediate face velocity is made divergence free via

$$\mathbf{u} = \mathbf{u}^* - \Delta t \nabla p. \tag{3}$$

where the cell centered pressure values are calculated by solving a Poisson equation of the form

$$\nabla^2 p = \nabla \cdot \mathbf{u}^* / \Delta t. \tag{4}$$

This equation is solved by assembling a symmetric system of linear equations, one for each MAC grid cell (that contains fluid) with the pressure defined at the cell center. In the case of water, we set Dirichlet $p = 1$ boundary conditions in air cells. A Neumann boundary condition implies that the pressure derivative at a cell face is zero, and thus the intermediate velocity is not modified as can be seen in equation 3.

We found that thin films of water can quickly compress and lose mass against thin solid objects if one is not careful in how the boundary conditions are handled. In fact, correctly handling the boundary conditions is of upmost importance for mass conservation in general, as discrepancies between the fluid and object velocity cause fluid to flow into or out of an object losing or gaining mass respectively. Our method for handling this is one of the key observations and contributions of this paper. First, we note that the velocity we compute for the fluid during the divergence free projection will be used in the *next* time step, and thus we need to make this commensurate with what the solid will do in the *next* time step. In order to do this, we calculate the size of the next fluid time step, evolve the solid object forward in time by the size of this time step allowing the solid to take as many substeps as it needs to remain

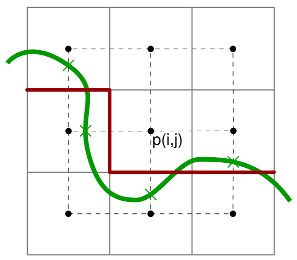

Figure 5: Neumann boundary conditions (denoted in bold red) are enforced at a cell face if the ray between two adjacent cell centers (where pressures are defined) intersects an object.

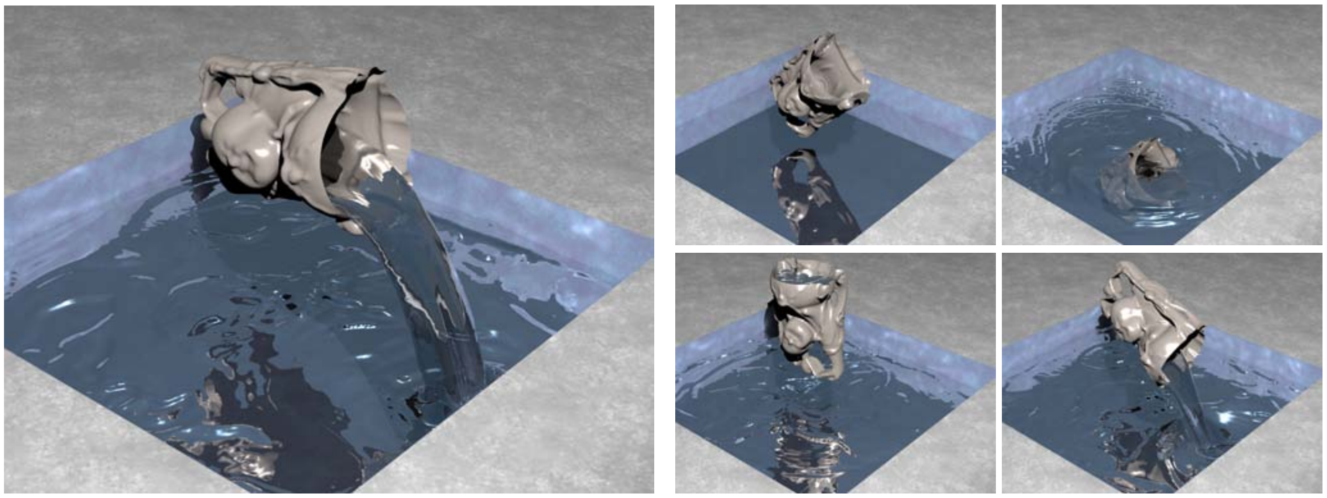
Figure 6: A rigid kinematically controlled "Buddha cup" dipped, filled and poured out (192^3 effective resolution octree, 60K triangles in the rigid cup).

accurate and stable, calculate an *effective velocity* for each node in the solid by dividing its positional change by the size of the next fluid time step, and finally rewind the solid to its current position at the end of the current fluid time step. Now the *effective velocity* represents exactly what the solid will do in the next time step, and we use Neumann boundary conditions to force the fluid to move in exactly this manner allowing for excellent resolution of thin films of water colliding against cloth and thin shells. We cast rays from a cell center to the six neighboring cell centers to see if an object cuts through the line segment connecting the pressures as shown in Figure 5. And if so, we set a Neumann boundary condition at the cell face and set the constrained velocity there equal to the appropriate component of the *effective velocity* of the object. Then the divergence is computed in the standard fashion, equation 4 is solved, and the results are used in equation 3.

A rather common difficulty with simulating highly deformable thin objects such as cloth in a fluid flow is that the cloth folds over on itself and pockets of fluid get separated from the flow. These are simple to identify by performing a flood fill algorithm over the fluid cell centers using the Neumann and Dirichlet conditions as the fill boundaries. If any region is surrounded entirely by Neumann boundary conditions, then the coefficient matrix assembled using equation 4 has a null space corresponding to the vector of all 1's and is not invertible. However, there is a version of the conjugate gradient algorithm that can be applied to this matrix, if we first enforce the compatibility condition [Peyret and Taylor 1983]. This is enforced independently in each region that has a null space using the area and velocity of the faces on the boundary to calculate the net flow per unit area into or out of the region. Then for each boundary face, we use this and the face area to obtain new temporary velocities that enforce no net flow across the region boundary. Finally, we solve for the pressure and make this region divergence free.

4.4 Water

We simulate water using the particle level set method of [Enright et al. 2002] with $\phi \leq 0$ denoting water and $\phi > 0$ denoting air. Since we only solve for velocity values in the water, each time step we extrapolate the nodal velocities across the interface into a 3-5 grid cell band to obtain velocity boundary conditions. To do this, we first order all the grid cells in the band based on their values of ϕ, noting that this ordering is only valid after reinitializing ϕ to be a signed distance function (see section 4.4.1). Then we solve the vector equation $\nabla\phi \cdot \nabla\mathbf{u} = 0$ for the nodes in ϕ increasing order. To prevent velocities from leaking across objects, we rule out neighbor values that are not *visible*. It is possible that some points have no *visible* neighbors, and we temporarily label these points *invalid*. After extrapolation is complete, all *invalid* points are given *valid* values as explained in section 3.

4.4.1 Level Set Method

[Enright et al. 2005] showed that the particle level set method relies primarily on the particles for accuracy whereas the role of the level set is to provide connectivity and smoothness. Thus, they showed that high order accurate level set advection could be replaced with a semi-Lagrangian characteristic based scheme without adversely affecting the accuracy. The level set is defined at the grid nodes, and thus we trace the same semi-Lagrangian rays as for velocity advection. When gathering the 8 values for trilinear interpolation, we replace nodes that are not *visible* with values averaged from a subset of the other 7 grid nodes of the cell whenever possible. Each point that needs to be redefined first looks to see if any of its one ring connected neighbors (in that 8 grid node cell) are *visible* to the base interpolation point. If so, they are averaged to obtain a new value for the node in question. Otherwise, we check and average the three 2 ring neighbors, or if that fails we consider the single 3 ring neighbor. If the process fails, there are no *visible* nodes in the cell and the point in question cannot be updated. We mark this node's ϕ value *invalid* and fix it in a postprocess (see section 3).

The level set is maintained to be a signed distance function using the fast marching method (see e.g. [Osher and Fedkiw 2002]). Typically, the nodes adjacent to the water interface are found by checking for sign changes between neighbors in the Cartesian grid directions (or along edges in the octree grid). We add to this list any node with $\phi \leq 0$ that has a neighbor that is not *visible*, and subtract from this list any node with $\phi > 0$ that does not contain a *visible* neighbor with $\phi \leq 0$. These last two adjustments ensure that an interface exists up against the solid object, and that water does not have influence across the thin triangulated surface. Typically, each node in this list is given an initial ϕ value by considering how far it is from the interface in each of the Cartesian grid directions that have a sign change. However, if the current node has $\phi > 0$, we ignore directions where the neighbor is not *visible*. And if $\phi \leq 0$, we use the minimum between the distance to the solid and $|\phi|$ in direc-

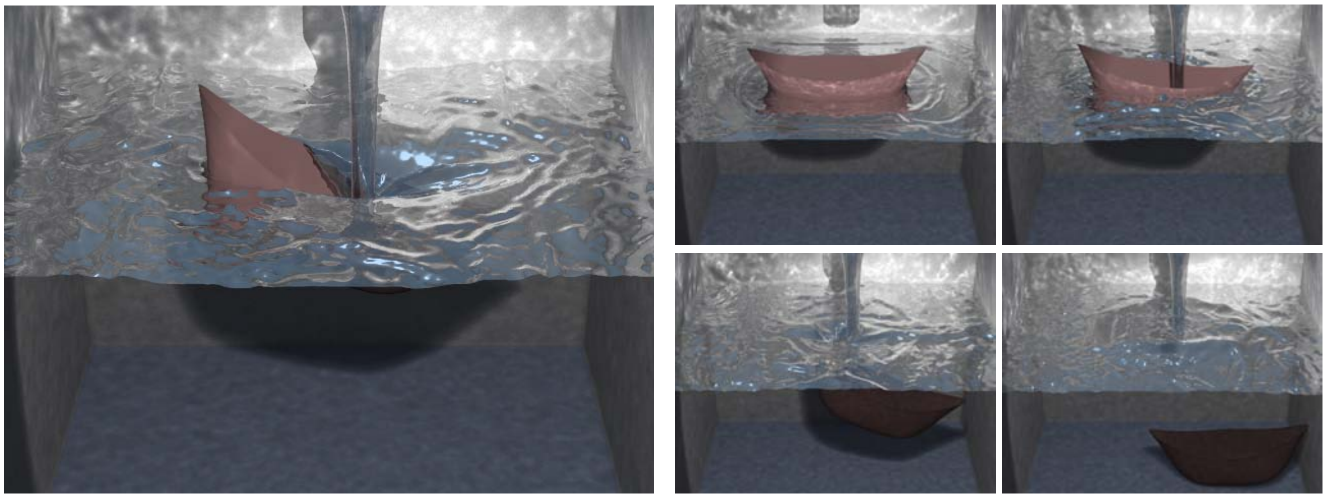
Figure 7: A full dynamic simulation of a rigid body shell two way coupled with water. The boat is heavier than the water, but retains buoyancy due to Archimedes' principle (effectively replacing displaced water with the air in its hull). Filling its hull with water causes it to sink, until it dynamically collides with the ground. ($148 \times 148 \times 111$ uniform grid, 2.5K triangles in the dynamically simulated rigid boat)

tions that are not *visible*. This last adjustment prevents water from incorrectly sticking to objects. After initialization, we employ the fast marching method in the usual fashion ruling out neighbors that are not *visible* when updating a given point (similar to extrapolation of velocity values).

4.4.2 Particle Level Set Method

Negative particles need to collide with solid objects to prevent water from leaking through those objects, and we collide them using a collision distance that is preassigned to each particle by drawing a random number between $.1\Delta x$ to Δx. To collide a particle with an object, we find the closest point on the object and compute the object normal at that location. We would like the particle to be at least its collision distance away from the object, and if it is not we move it in the normal direction by the required amount. If the particle intersects any object during this move we either delete it or just don't move it. We found that properly colliding negative particles against objects significantly improves the ability to properly resolve thin films of fluid against an object.

The particle velocity is determined by casting rays to the 8 neighboring nodal velocities in the same fashion as discussed above for the base of a semi-Lagrangian ray (see section 4.1). For negative particles that are closer than their collision distance to the object, we clamp the normal component of their velocity to be at least that of the object so that they do not get any closer to it. [Enright et al. 2005] showed that second order accurate particle evolution was quite important, especially when using the semi-Lagrangian method for level set advection. To achieve this, we first evolve the particles forward in time robustly colliding against the (stationary) object. Then we interpolate a new velocity at this location and average it with the original velocity to get a second order accurate velocity, before moving the particle back to its original location. For negative particles, we clamp the normal component of this averaged velocity if they are either within their collision distance to the object or they collided with the object when they were originally evolved.

Before moving each particle with its second order accurate velocity, we check for intersections between the particle center and the *moving* object. We delete positive particles that intersect the object, but attempt to adjust negative particles using the triangle they intersect. With the particle and triangle in their initial position, we record which side of the triangle the particle is on using the triangle normal. Then with the particle and triangle in their final position, we move the particle normal to the triangle so that it is on the same side as before and offset by its collision distance in the normal direction. Finally, we check this new particle path against the moving object and delete the particle if it still intersects the object. After advection, all negative particles are adjusted to be at least their collision distance away from the object as discussed above.

After updating both the level set and the particles, we modify the level set values using the particles. This is done in a collision aware fashion using only *visible* particles. The final step in the particle level set method is to adjust the radius of the particles based on the values of ϕ, and possibly delete particles that have escaped too far from the interface. This is accomplished by evaluating ϕ at the center of the particle in the same fashion as is done at the base of a semi-Lagrangian ray. Periodically, every 10-20 frames, particles are reseeded to get a better representation of the interface. Initially, this is performed disregarding the object altogether (for efficiency). Then as a postprocess, we evaluate ϕ at the center of the particle and delete particles with the wrong sign.

5 Cloth and Thin Shell Simulation

The novelty of our method lies both in the treatment of the fluid and in the interaction between the fluid and the solid, not in the simulation of the solids themselves. Thus, the solid simulation can be treated as a black box. This is a significant feature of our approach as one can use their favorite simulation technology for the solid object independent of the fluid solver and the solid/fluid two way coupling algorithm. All that is required is positions of the nodes of the triangulated surface at discrete time intervals, and from this we can calculate a velocity for each point assuming that it is piecewise constant between fluid time steps. When velocities within a triangle are required, we interpolate using the barycentric coordinates. For rigid body simulation, we use the method of [Guendelman et al. 2003], although our examples require no technology beyond that in [Hahn 1988; Moore and Wilhelms 1988]. For cloth, we use the basic cloth model from [Bridson et al. 2003] including their bending formulation, and the self-collision algorithm of [Bridson et al. 2002].

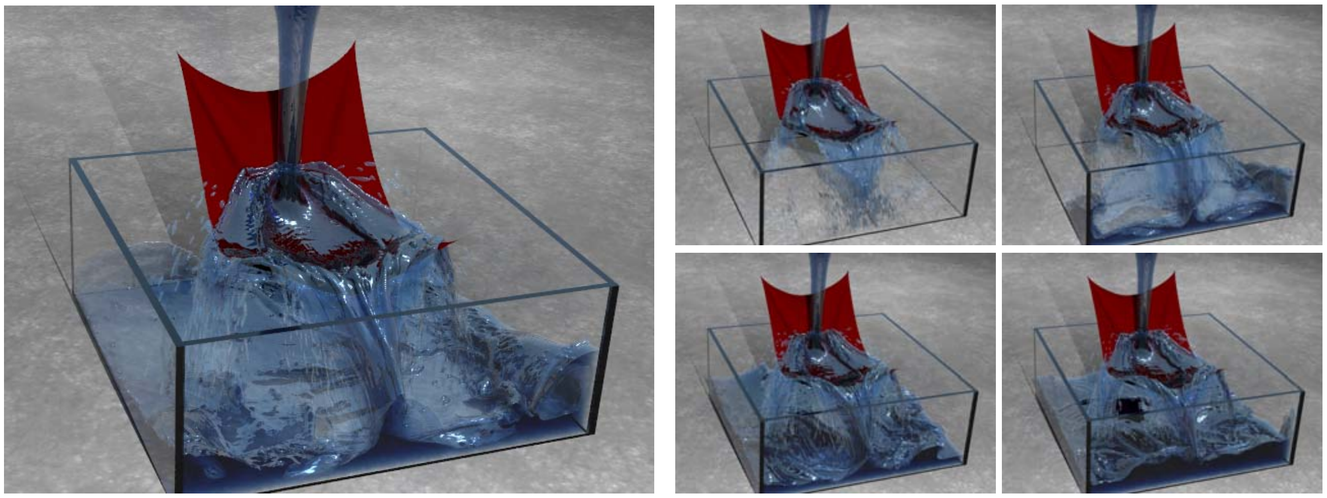
Figure 8: Two way coupled water flows over cloth (suspended at four corners), demonstrating that thin objects can support a sheet of water without leaks (200^3 effective resolution octree grid, 30K triangles in the cloth).

5.1 Coupling to the Fluid

The traditional method for coupling fluids and solids is to use the solid to prescribe velocity boundary conditions on the fluid, and the fluid to provide force boundary conditions on the solid [Benson 1992]. And as mentioned in section 4.3, it is important to coerce the fluid to move with the velocity of the solid so that thin films and sheets can be supported with little to no mass loss. For compressible fluids, one typically uses the pressure to provide a force on the surface of the solid, but for incompressible flow the pressure can be both stiff and noisy as compared to the velocity field, as discussed in [Fedkiw 2002]. Thus, we propose a new method that allows one to obtain forces on a thin deformable or rigid solid.

As in section 4.3, we cast rays from each cell center to the 6 neighboring cell centers to see if an object cuts through the line segment connecting the pressures as shown in Figure 5. And if so, we copy the local solid object velocity to the corresponding cell face in the usual fashion. In addition, we multiply the area of the face times the surface density of the solid to calculate a mass, m, for the cell face. Then we divide this mass by the local control volume which amounts to half of each neighboring cell (divided up appropriately among the faces for octrees). For a uniform grid $V = \triangle x \triangle y \triangle z$. Finally, the density of the cell face is set to m/V. All other cell faces have their density set to the fluid density. Since this is now a variable density flow, we need to solve a variable coefficient Poisson equation as in [Nguyen et al. 2002] where the analog to equations 3 and 4 are

$$\mathbf{u}^{new} = \mathbf{u}^{old} - \Delta t \nabla p/\rho \tag{5}$$

and

$$\nabla \cdot (\nabla p/\rho) = \nabla \cdot \mathbf{u}^{old}/\Delta t \tag{6}$$

respectively.

Note that we do not move the solid with this fluid velocity as for example in [Carlson et al. 2004] and [Zhu and Peskin 2002], but only use this velocity calculation to provide forces to the solid. The key advantage is that we do not have to figure out how to model solid response due to buoyancy, collisions, elasticity, etc. on the fluid dynamics grid and can instead simulate the solid with any black box method including finite elements, masses and springs, one's favorite collision algorithms, implicit or semi-implicit time integration, etc. As an added bonus, we are not hindered by the resolution of the fluid dynamics grid when evolving the solid, and a coarse grid merely provides a smeared out fluid force while still allowing all other solid dynamics and motion to be modeled with independently high resolution.

To calculate the force on the solid (after solving equation 6), we first calculate the pressure difference across each cell face that was considered to be part of the solid. Then we transfer the pressure differences from the faces to the nodes assigning the nodal value to be the average of the up to four face values that may be adjacent to it. We then spread these nodal differences to a band of grid points near the solid using the same Gauss-Jacobi averaging used to validate *invalid* nodes. This entire process is carried out independently for the pressure differences in each of the three coordinate directions. Then trilinear interpolation can be used to interpolate the vector pressure difference to the barycenter of each triangle, and a dot product with the triangle normal gives the pressure difference normal to (or across) the triangle face. Finally, we multiply by the triangle area and normal to get the net force on the face. In the case of a rigid body, the force and torque are accumulated by accounting for all the triangles. In the case of cloth, one third of the net force on a triangle is distributed to each node.

Since Neumann boundary conditions are enforced across the thin triangulated surface when solving equation 4, the computed pressure may have large jumps and contain noise. This is because that pressure has to force the fluid to take on the velocity of the solid to better enforce mass conservation. In contrast, Neumann conditions are not enforced when solving equation 6, and the computed pressure acts to change the velocities corresponding to the solid in equation 5 (although we do not actually apply this velocity change directly to the solid). This ability to account for the fluid's influence on the solid while solving an elliptic equation for the pressure provides for a much more stable coupling mechanism and smoother pressure values.

6 Algorithm Summary

Sequentially, the overall algorithm proceeds as follows. First, we compute the intermediate velocity $\mathbf{u}^*$ using knowledge of the solid's position at both time n and time $n+1$. Then we use $\mathbf{u}^*$ for $\mathbf{u}^{old}$ solving equation 6 for the solid fluid coupling pressure. This pressure is used to compute forces on the solid as it is evolved from time $n+1$ to time $n+2$, and that motion is used to compute the

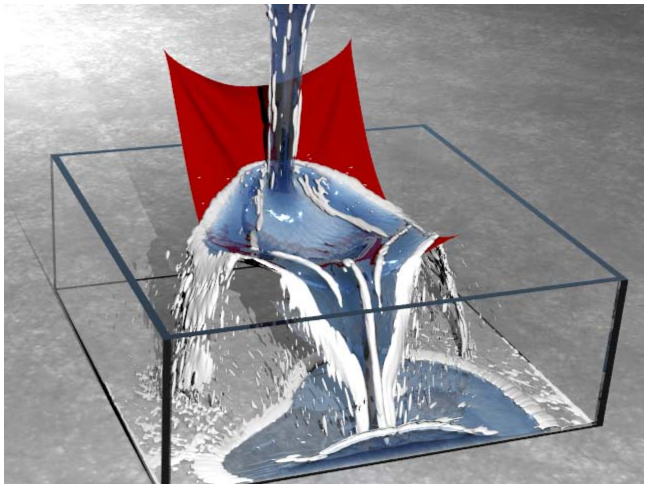
Figure 9: Illustration of the removed negative particles rendered as an opaque Lambertian surface (after proper blending). These particles have crossed over to the wrong side of the level set surface, but are too finely detailed to be properly represented by the fluid simulation grid.

effective velocity used to project the intermediate fluid velocity to be divergence free as discussed in section 4.3. Then the process is repeated.

7 Examples

We were able to simulate computational grids with effective resolutions as large as $256 \times 256 \times 192$ for the fluid and as many as 90k triangles for the rigid and deforming bodies using a 3 GHz Pentium 4. The computational cost ranged from 5 to 20 minutes per frame, and thus the longest examples took a couple of days. Rendering smoke and water against thin objects such as cloth poses complications as standard techniques interpolate density and level set values with stencils that intersect with thin bodies. Thus, we augmented a standard ray tracer with robust intersections and interpolations. That is, we used the same visibility based schemes used in simulation. This effectively removes visual artifacts caused by smoke and water showing through to the other side of objects, and air pockets showing through to the smoke and water side.

Figure 2 depicts a smoke stream flowing toward a suspended cloth curtain, and the two-way coupling generates interesting wrinkles and folds as well as smoke motion. Figure 4 depicts a kinematically controlled cup dipped, raised, and poured, demonstrating that our technique can model liquid behavior on both sides of a triangulated surface independently. Figure 6 shows similar behavior for more complex geometry. Full two-way coupling can be used with rigid shells as well, and Figure 7 shows a fully *dynamic simulation* of a boat floating until a stream of water sinks it. Figure 8 shows a stream of water flowing over a piece of cloth demonstrating full two-way coupling of cloth and water. Note specifically that the cloth supports the water (without leaks) and produces highly detailed thin water sheets flowing off the sides. For this example we augmented our ray tracer to additionally render the removed negative particles generated by the particle level set method, see Figure 9. This is accomplished by elongating the particles in the direction of their velocity vector, blending them together, and rasterizing both the particles and the level set function onto a fine octree grid before rendering. These particles added visually interesting splash and spray to the animation. Lastly, Figure 1 depicts a stream of water flowing onto a cloth curtain causing it to deform.

8 Conclusions and Future Work

We have presented a new computational algorithm for the coupling of incompressible flows to thin objects represented by moving triangulated surfaces. Examples were presented to demonstrate that this algorithm works well for one phase fluids such as smoke and for fluids with interfaces such as water. Moreover, it works with both rigid and deformable triangulated surfaces. Most importantly, our method prevents the leaking of material across the triangulated surface, while accurately enforcing the incompressibility condition in a one-sided fashion allowing for the interaction of thin films of water with highly deformable thin objects such as cloth.

Future work will include improvements in the way the removed particles are shaped, blended, and rendered. So far they were only used in Figures 8 and 9. In addition, it would be interesting to model the absorption of water by the cloth changing both its appearance and simulation properties. Also interesting, but not currently modeled, is the adhesion of water to the cloth.

9 Acknowledgements

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, ARO DAAD19-03-1-0331, NSF ITR-0121288, NSF DMS-0106694, NSF ACI-0323866, NSF IIS-0326388 and NSF ITR-0205671. We'd also like to thank Mike Houston, Christos Kozyrakis, Mark Horowitz, Bill Dally and Vijay Pande for computing resources.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. SIGGRAPH 98*, 1–12.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 862–870.

BARAFF, D. 1993. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10, 292–352.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. SIGGRAPH 94*, 23–34.

BENSON, D. 1992. Computational methods in lagrangian and eulerian hydrocodes. *Comput. Meth. in Appl. Mech. and Eng. 99*, 235–394.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 28–36.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 377–384.

CHEN, J., AND LOBO, N. 1994. Toward interactive-rate simulation of fluids with moving obstacles using the navier-stokes equations. *Computer Graphics and Image Processing 57*, 107–116.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 604–611.

COHEN, J. M., AND MOLEMAKER, M. J. 2004. Practical simulation of surface tension flows. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 736–744.

ENRIGHT, D., NGUYEN, D., GIBOU, F., AND FEDKIW, R. 2003. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.*, no. FEDSM2003–45144, ASME.

ENRIGHT, D., LOSASSO, F., AND FEDKIW, R. 2005. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures 83*, 479–490.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 441–448.

FEDKIW, R., STAM, J., AND JENSEN, H. 2001. Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001*, 15–22.

FEDKIW, R. 2002. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys. 175*, 200–224.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3, 708–715.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proc. of ACM SIGGRAPH 2001*, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graph. Models and Image Processing 58*, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Computer Graphics International 1997*, 178–188.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proc. of SIGGRAPH 97*, 181–188.

GÉNEVAUX, O., HABIBI, A., AND DISCHLER, J.-M. 2003. Simulating fluid-solid interaction. In *Graphics Interface*, 31–38.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRODER, P. 2003. Discrete shells. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 62–67.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3, 871–878.

HADAP, S., AND MAGNENAT-THALMANN, N. 2001. Modeling Dynamic Hair as a Continuum. *Comput. Graph. Forum 20*, 3.

HAHN, J. K. 1988. Realistic animation of rigid bodies. *Comput. Graph. (Proc. SIGGRAPH 88) 22*, 4, 299–308.

HONG, J.-M., AND KIM, C.-H. 2003. Animation of bubbles in liquid. *Comp. Graph. Forum (Eurographics Proc.) 22*, 3, 253–262.

HOUSTON, B., WIEBE, M., AND BATTY, C. 2004. Rle sparse level sets. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

IVERSEN, J., AND SAKAGUCHI, R. 2004. Growing up with fluid simulation on "the day after tomorrow". In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

KANG, M., FEDKIW, R., AND LIU, X.-D. 2000. A boundary condition capturing method for multiphase incompressible flow. *J. Sci. Comput. 15*, 323–360.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *Computer Graphics (Proc. of SIGGRAPH 90)*, vol. 24, 49–57.

KONDOH, N., KUNIMATSU, A., AND SASAGAWA, S. 2004. Creating animations of fluids and cloth with moving characters. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

LAMORLETTE, A., AND FOSTER, N. 2002. Structural modeling of natural flames. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 3, 729–735.

LI, Z., AND LAI, M.-C. 2001. The immersed interface method for navier-stokes equations with singular forces. *J. Comput. Phys. 171*, 822–842.

LING, L., DAMODARAN, M., AND GAY, K. 1996. Aerodynamic force models for animating cloth motion in air flow. In *The Visual Computer*, 84–104.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 457–462.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. (SIGGRAPH Proc.)*.

MIHALEF, V., METAXAS, D., AND SUSSMAN, M. 2004. Animation and control of breaking waves. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 315–324.

MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animation. *Comput. Graph. (Proc. SIGGRAPH 88) 22*, 4, 289–298.

MULLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 154–159.

MULLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids. *J. Comput. Anim. and Virt. Worlds 15*, 3–4 (July), 159–171.

NGUYEN, D., FEDKIW, R., AND JENSEN, H. 2002. Physically based modeling and animation of fire. In *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 29, 721–728.

NIXON, D., AND LOBB, R. 2002. A fluid-based soft-object model. *IEEE Comput. Graph. Appl. 22*, 4, 68–75.

NOH, W. 1964. *CEL: A time-dependent, two-space-dimensional, coupled Eulerian-Lagrange code*. Academic Press, New York, 117–179.

OSHER, S., AND FEDKIW, R. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag. New York, NY.

PESKIN, C. 1972. Flow patterns around heart valves: A numerical method. *J. Comput. Phys. 10*, 252–271.

PESKIN, C. 2002. The immersed boundary method. *Acta Numerica 11*, 479–517.

PEYRET, R., AND TAYLOR, T. D. 1983. *Computational methods for fluid flow*. Springer-Verlag. New York.

PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. 2003. Particle–based simulation of fluids. In *Comp. Graph. Forum (Eurographics Proc.)*, vol. 22, 401–410.

RASMUSSEN, N., NGUYEN, D., GEIGER, W., AND FEDKIW, R. 2003. Smoke simulation for large scale phenomena. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 703–707.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directible photorealistic liquids. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 193–202.

STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH 99*, 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 724–731.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 3, 716–723.

WEI, X., ZHAO, Y., FAN, Z., LI, W., YOAKUM-STOVER, S., AND KAUFMAN, A. 2003. Blowing in the wind. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 75–85.

WEJCHERT, J., AND HAUMANN, D. 1991. Animation Aerodynamics. *Comput. Graph. 25*, 4, 19–22.

WIEBE, M., AND HOUSTON, B. 2004. The tar monster: Creating a character with fluid simulation. In *SIGGRAPH 2004 Sketches & Applications*, ACM Press.

YNGVE, G., O'BRIEN, J., AND HODGINS, J. 2000. Animating explosions. In *Proc. SIGGRAPH 2000*, vol. 19, 29–36.

ZHU, L., AND PESKIN, C. 2002. Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. *J. Comput. Phys. 179*, 452–468.