Multiple Interacting Liquids

Frank Losasso*
Stanford University
Industrial Light + Magic

Tamar Shinar*
Stanford University

Andrew Selle*
Stanford University
Intel Corporation

Ronald Fedkiw†
Stanford University
Industrial Light + Magic

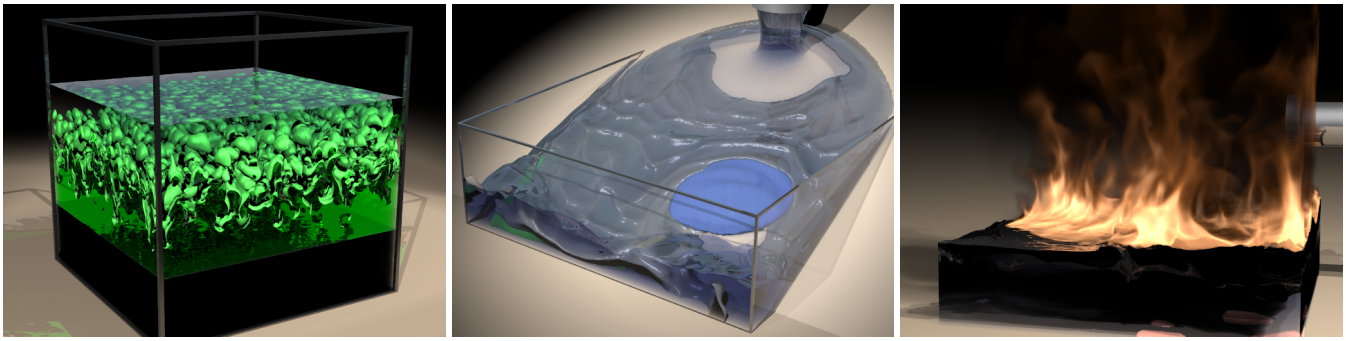


Figure 1: (Left) Rayleigh-Taylor instability, 4 phases. (Center) liquids of varying viscosity, 5 phases. (Right) burning oil in water, 4 phases.

Abstract

The particle level set method has proven successful for the simulation of *two* separate regions (such as water and air, or fuel and products). In this paper, we propose a novel approach to extend this method to the simulation of as many regions as desired. The various regions can be liquids (or gases) of any type with differing viscosities, densities, viscoelastic properties, etc. We also propose techniques for simulating interactions between materials, whether it be simple surface tension forces or more complex chemical reactions with one material converting to another or two materials combining to form a third. We use a separate particle level set method for each region, and propose a novel projection algorithm that decodes the resulting vector of level set values providing a "dictionary" that translates between them and the standard single-valued level set representation. An additional difficulty occurs since discretization stencils (for interpolation, tracing semi-Lagrangian rays, etc.) cross region boundaries naively combining non-smooth or even discontinuous data. This has recently been addressed via ghost values, e.g. for fire or bubbles. We instead propose a new paradigm that allows one to incorporate physical jump conditions in data "on the fly," which is significantly more efficient for multiple regions especially at triple points or near boundaries with solids.

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

Keywords: multiphase fluids, liquids, gases, level sets

1 Introduction

Earlier works on fluid simulation focused on single phase flows such as smoke [Foster and Metaxas 1997b; Stam 1999; Fedkiw et al. 2001] or free surface flows such as water [Foster and Fedkiw 2001; Enright et al. 2002]. More recently, researchers have considered more complex phenomena including fire [Lamorlette and Foster 2002; Nguyen et al. 2002], bubbles [Hong and Kim 2005], viscoelasticity [Goktekin et al. 2004], etc. Although the level set method allows for the simulation of two distinct fluids, such as fuel and products [Nguyen et al. 2002] or water and air [Hong and Kim 2005], it does not handle the complex phenomena associated with the interactions of more than two fluids. In this paper, we propose a novel method that allows one to simulate multiple (more than two) liquids (and gases), including complex interactions between the different fluids.

A few researchers have begun to tackle the difficulties associated with using multiple level sets (without particles) to represent multiple regions. One approach is to use a different level set for each region as in [Merriman et al. 1994; Ruuth 1998; Smith et al. 2002]. Each level set is independently evolved forward in time leading to contradictions in the representation of the interface, which are resolved via projection (or slowly, via a penalty method as in [Zhao et al. 1996]) eliminating points that have been classified as inside more than one region or not inside any region. The other main approach uses n level sets to represent up to 2^n regions [Vese and Chan 2002]. For example, two level sets can be used to represent four regions classified via all possible sign combinations (i.e. "++", "+−", "−+" and "−−"). This approach intrinsically removes the need for projection, but typically suffers from biasing artifacts especially where more than two regions intersect. So while it is quite useful in computer vision especially when there are *many* regions, it has not enjoyed similar success in physics-based applications where the number of distinct materials is typically small enough that it is not inefficient to use a separate level set for each allowing for a non-biased simulation of the underlying physics. In fact, level set methods are typically only applied in a lower dimensional band near the interface making them cheap as compared to the physical equations that need to be solved everywhere. Moreover, regardless of the number of level sets used, the particle level set method still requires one set of particles for each region, so its cost remains unchanged. Finally, we note that all these approaches are predated by [Gascuel 1993], which proposed a method for removing overlaps of implicitly represented deformable objects, although gaps between objects were not addressed making the work inapplicable to fluids where vacuum regions need to be properly addressed.

At each point in the domain, we have a vector $\vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \ldots, \phi_n(\vec{x}))$. Since the individual level set functions (the ϕ_i's)

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

will generally give contradictory geometric information, a consistent approach to interpreting the vector-valued level set function is needed. We do this by creating a level set "dictionary" that translates between the vector $\vec{\phi}$ and the traditional single level set representation at each point in the domain. Our novel projection method makes this translation straightforward for both theoretical and practical purposes (e.g. allowing the incorporation of previous level set simulation techniques). Moreover, our method is purely geometric and thus does not interfere with the underlying physics. It also preserves the signed distance property of the various level set functions (unlike for example [Merriman et al. 1994]).

Our method provides for the straightforward simulation of multiple liquids (and air) with varying densities, viscosities, or viscoelastic properties. We also consider complex interactions between fluids such as surface tension forces and reactions (e.g. the burning of a premixed fuel as in [Nguyen et al. 2002]). Such interactions typically involve discontinuous material properties across the interface, e.g. pressure jumps due to surface tension. [Nguyen et al. 2002] and [Hong and Kim 2005] advocated using the ghost fluid method (GFM) to avoid the visual errors associated with nonphysically smearing out these discontinuities. We propose a novel paradigm that automatically detects when discontinuous information is combined across (any number of) interfaces, computes jump conditions and ghost values "on the fly," and returns appropriate values. This reduces the memory requirements associated with storing ghost values for multiple region interactions, and furthermore makes the implementation of the algorithms straightforward. We note that it also simplifies the treatment of complex solid objects (see e.g. [Guendelman et al. 2005]).

2 Previous Work

Besides the works already mentioned above, earlier computer graphics research on the Navier-Stokes equations includes [Kass and Miller 1990; Chen and Lobo 1994; Foster and Metaxas 1996; Foster and Metaxas 1997a]. There has also been work on explosions [Neff and Fiume 1999; Yngve et al. 2000; Feldman et al. 2003; Rasmussen et al. 2003], flow on surfaces [Stam 2003], chemically reacting gases [Ihm et al. 2004], octree implementations [Losasso et al. 2004], RLE implementations [Houston et al. 2006], tetrahedral meshes [Feldman et al. 2005] hybridized vortex particle approaches [Selle et al. 2005] and sand [Zhu and Bridson 2005]. Various authors have also addressed viscosity [Carlson et al. 2002; Rasmussen et al. 2004; Hong and Kim 2005], surface tension [Enright et al. 2003; Hong and Kim 2003; Losasso et al. 2004; Hong and Kim 2005] and fire [Stam and Fiume 1995; Lamorlette and Foster 2002; Nguyen et al. 2002; Melek and Keyser 2005]. Some of the most recent interesting areas include control [Treuille et al. 2003; McNamara et al. 2004; Fattal and Lischinski 2004; Rasmussen et al. 2004; Mihalef et al. 2004; Shi and Yu 2005], solid fluid coupling [Carlson et al. 2004; Guendelman et al. 2005; Wang et al. 2005; Losasso et al. 2006], and SPH [Premoze et al. 2003; Keiser et al. 2005]. In fact, [Müller et al. 2005] tackles the problem of interacting multiple fluids from the SPH standpoint.

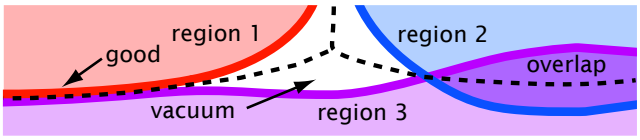


Figure 2: Each of the three regions is independently evolved in time, after which the interface locations do not agree. There are vacuums where all ϕ_i are positive, and overlaps where more than one ϕ_i is negative. The dotted black line shows the new interface locations after our projection step.

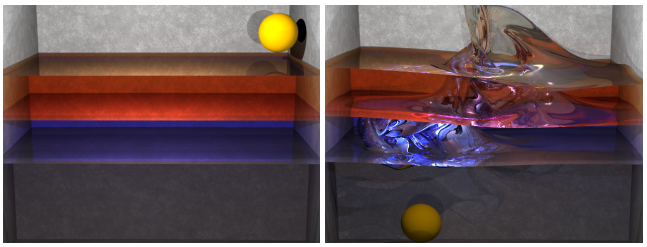


Figure 3: A kinematically controlled sphere splashing into a multi-layer pool ($300 \times 300 \times 200$ grid, 4 phases).

3 Multiple Level Sets

Each level set function is independently evolved in time, after which the interface locations do not agree (because of numerical errors) as shown for example in Figure 2. We propose a novel method for fixing the level set functions removing overlaps and vacuums while preserving an accurate interface location.

3.1 Projection Method

We first make the following observations about an arbitrary vector $\vec{\phi}$ of level set values at a point $\vec{x}$. (O1) If ϕ_j is the smallest element, $\vec{x}$ is in region j. This assigns $\vec{x}$ to the region it is deepest inside when it is inside more than one region (overlap), or the region it is closest to when it is outside every region (vacuum). (O2) If O1 holds and ϕ_k is the second smallest element, only ϕ_j and ϕ_k are needed to locally represent the interface. Basically, $\vec{x}$ is in region j, and the closest point on an interface lies between region j and region k. Region k is the region $\vec{x}$ is closest to not counting the region it is in.

Given these observations, we desire the following properties for numerical robustness and backward compatibility with the standard single level set function for two phases. (P1) If ϕ_j is the smallest element, it is the only negative element and its magnitude represents the distance to the interface. This is consistent with observation 1, but also makes ϕ_i a signed distance function in region i for all i. Moreover, it removes overlaps since only one ϕ_i is negative, and removes vacuums since the smallest ϕ_i *is* negative. (P2) If P1 holds and ϕ_k is the second smallest element, $\phi_k = -\phi_j$. This is consistent with observation 2, and it makes the level set for the region a point is closest to but not inside a signed distance function as well. Moreover, all other ϕ_i are positive and bigger than ϕ_k and not relevant.

These observations and properties are consistent with the standard single level set function methodology. A standard single level set

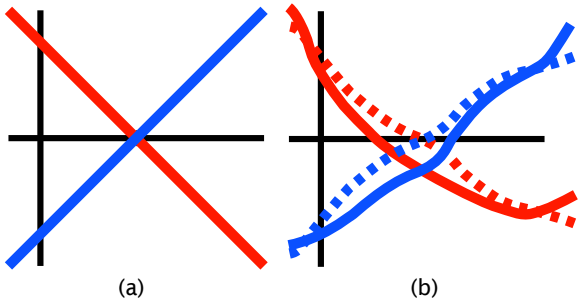


Figure 4: (Left) Two level sets initialized so that properties 1 and 2 hold. (Right) After evolving in time, we obtain the solid lines with overlap (both negative in the middle). The dotted lines show an example result after projection. Not only has the overlap been removed, but the interface location is preserved.

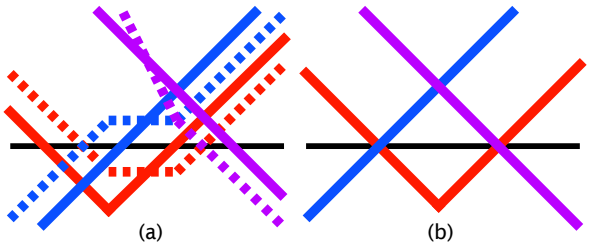


Figure 5: (Left) The solid lines have overlap on the left (two ϕ_i negative) and a vacuum on the right (all ϕ_i positive). The average of the smallest two level sets at any point is subtracted from $\vec{\phi}$ to obtain the dotted lines. Not only has the overlap and vacuum been removed, but the smallest ϕ_i is preserved and negative at each point preserving interface locations and inside/outside information. (Right) Results after reinitializing each level set to a signed distance function.

function ϕ can be broken into two separate functions ϕ_1=ϕ and $\phi_2 = -\phi$, and be shown to satisfy the above observations and properties. This readily gives us a dictionary that translates between $\vec{\phi}$ and ϕ. That is, once we take an arbitrary level set vector $\vec{\phi}$ and project it to satisfy properties 1 and 2, we can use ϕ_j and ϕ_k as if they were ϕ_1 and ϕ_2 in the appropriate order. The only discrepancy lies in how the exact interface is handled where $\phi_j = \phi_k = 0$. For the standard level set function, this is equivalent to $\phi_1 = \phi_2 = \phi = 0$ and is typically nominally assigned to the negative level set, i.e. we define the regions via $\phi \leq 0$ and $\phi > 0$. This is equivalent to assigning the point to region 1 when ϕ_1 and ϕ_2 are both zero. To extend this to multiple level sets, we assign a point where both ϕ_j and ϕ_k are zero to region j or k depending on whether $j < k$.

We illustrate our method in one spatial dimension. Figure 4a shows two level set functions that satisfy properties 1 and 2, and Figure 4b shows a property violating version after evolving in time. Based on property 1, the interface location is defined as the point where the minimum ϕ_i changes from one level set to the other. This is the location where the two level sets intersect in the figure, and we want our projection method to preserve the interface location to avoid biasing. Thus, our projection method *computes the average value of the two level set functions and subtracts this average from both of them.* At points where the two level sets intersect, their average equals their individual values, and thus subtracting off their averages sets them both to zero preserving the interface location. Otherwise, at points where one level set is larger than the other, subtracting their average makes them the same magnitude but opposite sign preserving the region a given point is inside. The result is shown as dotted lines in the figure. If both level sets are reinitialized to signed distance functions, we obtain the result shown in Figure 4a which satisfies all desired properties. *The same projection method can be generalized to an arbitrary vector of level sets by subtracting the average of the smallest two ϕ_i from all of the ϕ_i.* An example of this is shown in Figure 5.

Notably our method is unbiased preserving signed distance information. For example, consider Figure 4a and Figure 5b where the level sets are all signed distance functions to begin with. Because property 2 holds, the two smallest ϕ_i are equal and opposite in sign making their average identically zero at every point. Thus, subtracting the average leaves all the the ϕ_i unchanged preserving signed distance. This surprisingly simple algorithm has all the properties we desire. Notably, [Merriman et al. 1994] is similar in spirit to our own, but while they preserve the interface location and inside/outside information, they do not preserve signed distance thus introducing biasing into the algorithm.

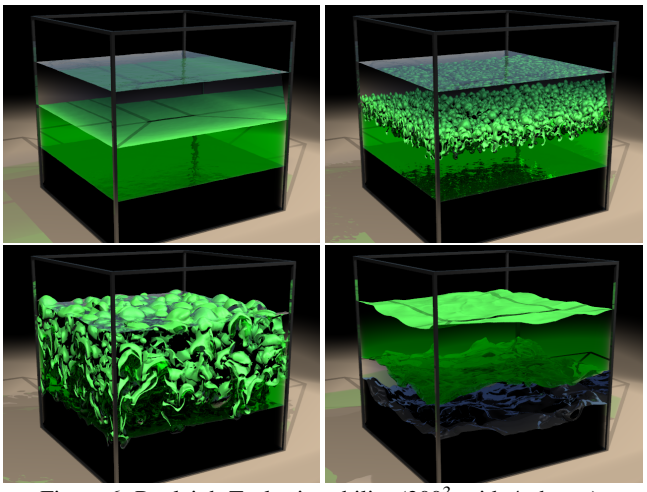
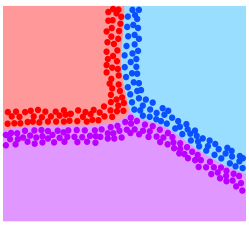


Figure 6: Rayleigh-Taylor instability (300^3 grid, 4 phases).

For two regions (hence two level sets) the result produced by our method is identical to that of the traditional single level set (or particle level set) method. If the two level set functions at time n are negatives of each other, i.e. $\phi_1^n = -\phi_2^n$, after advection we still have $\phi_1^* = -\phi_2^*$ because they are evolved with the same method. Then projection leaves ϕ_1^* and ϕ_2^* unchanged, since their identically zero average is subtracted off. This is also true for more than two regions away from multiple junctions (e.g., triple points).

3.2 Particle Level Set Method



Each level set has an associated set of particles that are seeded near the boundary of its interior region as shown in the figure to the right. Following the standard particle level set algorithm, for each level set function we rebuild ϕ^- using that level set's particles, and rebuild ϕ^+ using all the particles from all other regions. For efficiency, we ignore particles that are far from the interface of the region in question. Typically, particles are used to correct the level set function both after advection and after reinitialization. We apply our projection method to every grid point after each of these particle correction steps. Then for each grid point, all geometric information can be computed from the level set function that is negative at that point. When level set values are needed in between grid points, we interpolate $\vec{\phi}$ to that location and apply our projection method on the fly to find the resulting negative ϕ_i. Note that the first projection step is important because reinitialization preserves the interface location of each level set individually, but not their intersections which correspond to the pre-projected interface locations. The second projection removes any numerical drift introduced by reinitialization, and we note that a method such as [Merriman et al. 1994] could not be used for this step because it does not preserve signed distance.

Advancing the incompressible velocity field to the next time step and particle advection are the most expensive parts of our algorithm. These steps depend mostly on the fluid volume and surface area, respectively, rather than the number of regions. The cost of advecting and reinitializing the level set functions remains fixed at twice the usual cost, since two level sets are updated locally near each interface instead of one. Thus, the cost of updating the level set function scales with the surface area just as in the standard single level set method regardless of the number of regions.

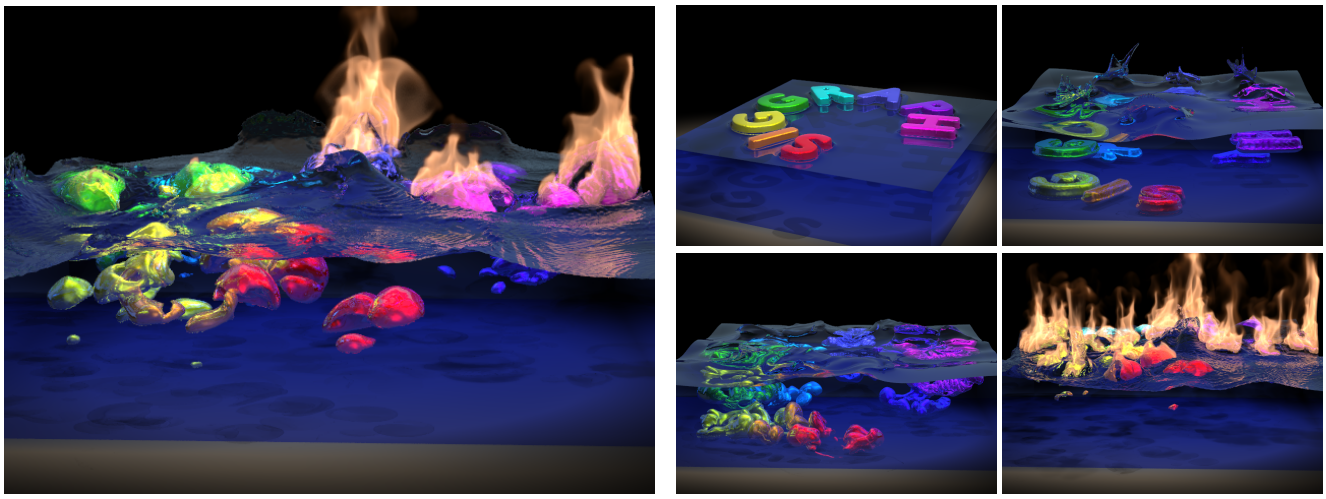


Figure 7: Viscous letters splash into a pool of water, then change into low density inviscid fuel bubbling up and burning when they hit the surface ($350 \times 200 \times 350$ grid, 10 phases).

4 Multiple Liquids

We model the fluids using the incompressible Navier-Stokes equations

$$\nabla \cdot \vec{u} = 0 \quad (1)$$

$$\vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} + \nabla p/\rho = (\nabla \cdot \tau)/\rho + \vec{f} \quad (2)$$

where $\vec{u} = (u, v, w)$ is the velocity, ρ is the density, τ is the viscous stress tensor, and $\vec{f}$ accounts for body forces, e.g. gravity, vorticity confinement, etc. For simplicity, we first consider the inviscid case. First, an intermediate velocity field $\vec{u}^\star$ is computed

$$(\vec{u}^\star - \vec{u}^n)/\Delta t + (\vec{u}^n \cdot \nabla)\vec{u}^n = \vec{f} \quad (3)$$

using a semi-Lagrangian advection scheme as in [Stam 1999]. Next, we compute the pressure via

$$\nabla \cdot (\nabla p/\rho) = \nabla \cdot \vec{u}^\star/\Delta t. \quad (4)$$

and use it to make the velocity field divergence free

$$(\vec{u}^{n+1} - \vec{u}^\star)/\Delta t + \nabla p/\rho = 0. \quad (5)$$

4.1 Poisson Equation

We follow the method of [Nguyen et al. 2002]. For multiple fluid regions, equation (4) is a Poisson equation with discontinuous coefficients. The equation is separable so we can consider each dimension independently. A standard second order accurate discretization of the left hand side in one spatial dimension at a grid node i is

$$\left(\beta_{i+1/2}(p_{i+1} - p_i)/\Delta x - \beta_{i-1/2}(p_i - p_{i-1})/\Delta x\right)/\Delta x \quad (6)$$

where $\beta = 1/\rho$. For inviscid flow, [Kang et al. 2000] showed that the flux in equation (4) is continuous across the interface satisfying

$$\beta^- p_x^- = \beta^+ p_x^+ \quad (7)$$

where the $-$ and $+$ superscripts represent values from different sides of the interface. Thus if ρ (and hence β) varies across the interface then so must p_x. Consider the case where an interface lies between nodes x_i and x_{i+1}. We define $\theta = |\phi(x_i)|/(|\phi(x_i)| + |\phi(x_{i+1})|)$ and approximate equation (7) with one-sided differences as

$$\beta^-(p_I - p_i)/(\theta\Delta x) = \beta^+(p_{i+1} - p_I)/((1-\theta)\Delta x) \quad (8)$$

and solve for the interface pressure $p_I = (\theta\beta^+ p_{i+1} + (1-\theta)\beta^- p_i)/(\theta\beta^+ + (1-\theta)\beta^-)$ which can be substituted into either the left or the right hand side of equation (8) to obtain $\hat{\beta}(p_{i+1} - p_i)/\Delta x$ where $\hat{\beta} = (\beta^-\beta^+)/(\theta\beta^+ + (1-\theta)\beta^-)$. Thus, the discontinuity between grid nodes i and $i+1$ is readily handled by replacing $\beta_{i+1/2}$ with $\hat{\beta}$ in equation (6). $\beta_{i-1/2}$ is treated similarly.

4.2 Viscosity

The viscous stress tensor for incompressible flow is $\tau = \mu(\nabla\vec{u} + (\nabla\vec{u})^T)$. As discussed in [Rasmussen et al. 2004], a spatially constant μ (within each region) implies that $\nabla \cdot \tau = \mu\Delta\vec{u}$. Renaming $\vec{u}^{n+1}$ in equation (5) to be $\vec{u}^{\star\star}$, we next solve the three systems of linear equations given by

$$\vec{u}^{\star\star\star} = \vec{u}^{\star\star} + \Delta t \nabla \cdot (\nu \nabla \vec{u}^{\star\star\star}) \quad (9)$$

where $\nu = \mu/\rho$. Note that we moved ρ under the divergence operator, under the assumption that it is spatially constant in each region. Since the viscosity to density ratio is discontinuous across the interface, we replace ν with $\hat{\nu}$ for differences that cross the interface in the same manner as β is adjusted to $\hat{\beta}$ when solving equation (4). Then, we again solve for the pressure and make the flow divergence free using $\vec{u}^{\star\star\star}$ in place of $\vec{u}^\star$ in equations (4) and (5).

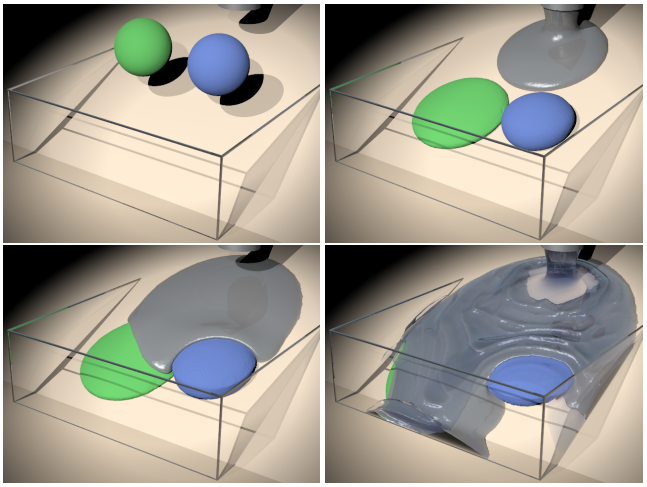


Figure 8: Different viscosity liquids interacting on an inclined plane. ($300 \times 150 \times 240$ grid, 5 phases).

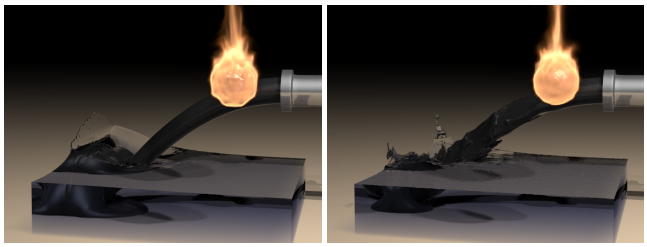


Figure 9: (Left) One way coupling from liquid to air. (Right) Two way coupling of liquid and air. Note the surface ripples and the unstable stream of liquid in the fully coupled simulation.

Each component of equation (9) should conserve momentum, so we require a unique flux between every two velocity values. The physically correct flux in the incompressible flow context is rather complicated (see its derivation in [Kang et al. 2000]). However, considering equation (9) in isolation admits a simple approximation which [Hong and Kim 2005] showed was sufficient for visual accuracy. In isolation, the flux is given by $\nu\nabla\vec{u}$, and we assume $\nu^-\nabla\vec{u}^- = \nu^+\nabla\vec{u}^+$ which is not actually true.[1] Instead, correction terms should be added for stencils that cross the interface, but these terms couple the u, v, and w diffusion equations together making them difficult to solve with a fully implicit method. For example, see [Rasmussen et al. 2004] where variable viscosity couples the three equations together.[2] They explicitly add correction terms before solving for the velocity implicitly. We could take a similar approach allowing for a larger time step than a fully explicit method, but it is still less efficient than a fully implicit method. [Kang et al. 2000] showed that the viscosity jump causes a jump in the pressure and its derivatives as well[3], but [Hong and Kim 2005] showed that these too can be ignored for graphical purposes.

4.3 Viscoelasticity

[Goktekin et al. 2004] incorporated viscoelastic effects by adding $(\mu_e/\rho)\nabla\cdot\varepsilon$ to the Navier-Stokes equations, where μ_e is the elastic modulus and ε is the elastic strain tensor evolved in time via $\varepsilon_t + \vec{u}\cdot\nabla\varepsilon = (\nabla\vec{u} + (\nabla\vec{u})^T)/2 - \varepsilon_t^{Plastic}$. They solved this last equation by first using semi-Lagrangian advection, and then incorporating the right hand side which is the total strain rate minus the plastic strain rate. [Irving 2007] points out that this ignores the rotation of the strain tensors, yielding incorrect results when the fluid rotates. Thus, [Irving 2007] proposes rotating the strain tensor by the curl of the velocity field after the advection step. This is accomplished by computing an explicit rotation matrix in the center of each cell, and using it to rotate the strain tensor also stored in the center of each cell. This has enabled high quality detailed viscoelastic fluid computations, see e.g. Figure 13.

5 Adding Air or Empty Regions

Often times, only the liquid region is of interest and the gas flow can be ignored. Our system allows for the standard treatment of this by setting an entire region to be empty, and subsequently extrapolating velocity into that region from the liquid regions and using Dirichlet boundary conditions during the pressure solve (see e.g.[Enright et al. 2002]). However, when the gas flow is important, our method trivially extends to simulate gas regions just as though they were other liquid regions. This allows for straightforward incorporation of smoke, fire, and even reactive gases as in [Ihm et al. 2004]. Besides empty regions and air regions, there is yet a third way to model

[1] [Kang et al. 2000], equation (30) gives the jump across the interface

[2] [Rasmussen et al. 2004], equations (5)-(7)

[3] [Kang et al. 2000], equations (19) and (32)-(34)

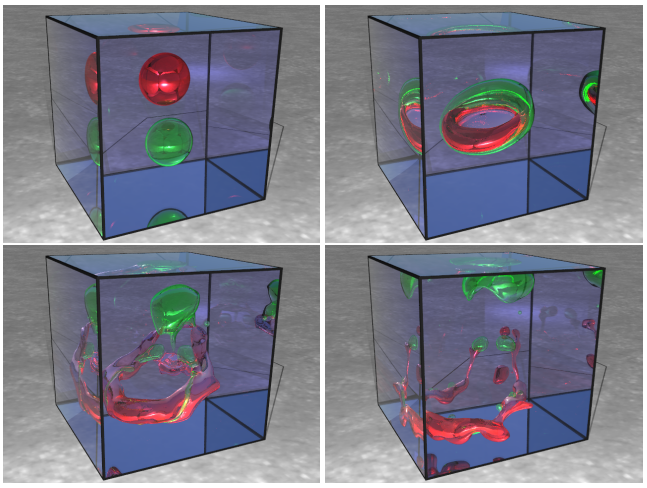
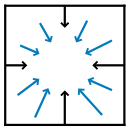


Figure 10: Two drops with high surface tension collide. Green has low density, red high density (350^3 grid, 3 phases).

non-liquid regions. The animator may wish to simulate air, but not have the air affect the liquid. This requires one way coupling from liquid to air, but not vice versa. Figure 9 shows one way coupling (left) as compared to two way coupling (right). One way coupling is accomplished by using extrapolated velocities from the liquid to the gas as boundary conditions for the liquid region, while gas advection is carried out normally. In addition, these extrapolated velocities are used to overwrite the gas velocity at any grid points that become liquid as the interface moves. The Poisson equation is first solved for the liquid region setting Dirichlet boundary conditions in the gas so that it has no effect (as is usual for empty regions), and then a second Poisson equation is solved for the gas using Neumann fixed velocity boundary conditions in the liquid so that it properly drives the gas.



In free surface flow, the air region is modeled as empty allowing it to vanish. Thus, characteristics coalesce as shown in the single grid cell in the figure to the right. Eventually, liquid rushes into the cell from all sides, and the air should disappear. However, air particles faithfully follow these characteristics ending up trapped in the center of the cell. Since this cell should be a sink for air, we simply delete the air particles as they approach the center of the sink. Note that these sinks are easily detected by finding local minima level set values in empty regions.

6 Surface Tension

The ghost fluid method (GFM) of [Fedkiw et al. 1999] uses the physically correct interfacial jump conditions to define ghost values for discontinuous quantities which are then incorporated into finite difference or interpolation stencils. [Hong and Kim 2005] used the GFM to discretize the jumps in pressure caused by surface tension effects, and [Hong 2005][4] showed that the method produces far better visual results than a smeared out delta function approach. Surface tension causes a jump in pressure across the interface equal to $\sigma\kappa$, where σ is a surface tension coefficient (defined pairwise for the regions) and $\kappa = -\nabla\cdot(\nabla\phi/|\nabla\phi|)$ is the interface curvature. Consider the case where an interface lies between x_i and x_{i+1}. We adjust p_{i+1} in equation (6) to account for the jump in pressure,

$$\left(\hat{\beta}((p_{i+1}+\sigma\kappa_\Gamma)-p_i)/\Delta x - \beta_{i-1/2}(p_i-p_{i-1})/\Delta x\right)/\Delta x$$

[4] page 36

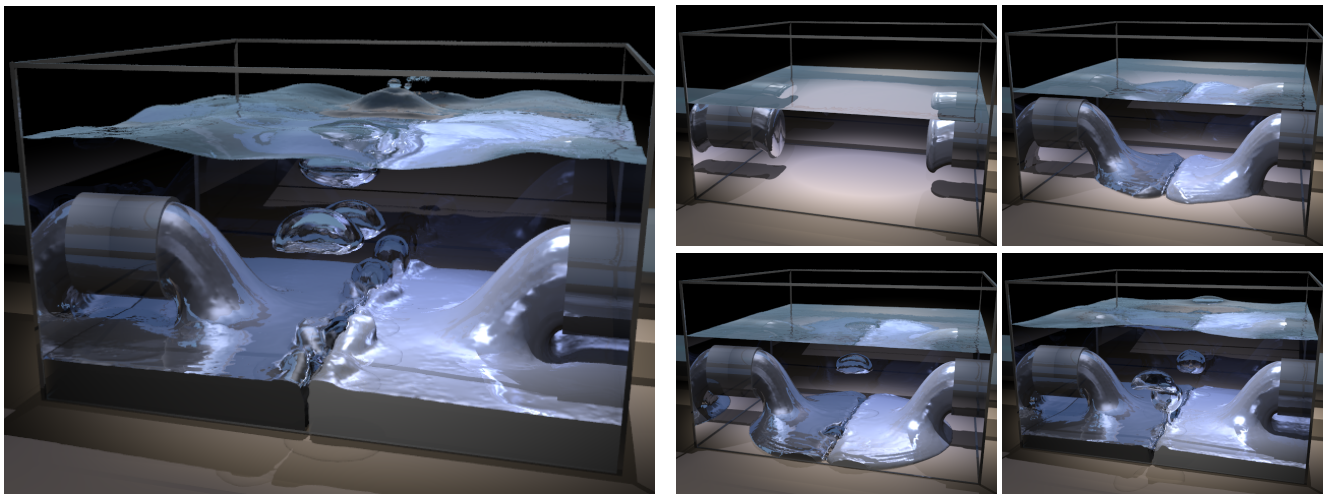


Figure 11: Two submerged liquids meeting and reacting to create air (150^3 grid, 4 phases).

where $\kappa_\Gamma = \theta\kappa_{i+1} + (1-\theta)\kappa_i$ is computed with respect to the region containing x_i, and $\beta_{i+1/2}$ has been replaced by $\hat{\beta}$ as explained in section 4.1. The $(\hat{\beta}\sigma\kappa_\Gamma)/\Delta x^2$ term can be moved to the right hand side, so that the resulting matrix is unaffected allowing for the use of fast symmetric linear system solvers such as the preconditioned conjugate gradient method.

7 Surface Reactions

Our method incorporates surface reactions allowing one material to turn into another. One example of this is the work on fire by [Nguyen et al. 2002], but our framework allows for a more generalized treatment, e.g. Figure 11 shows two materials (submerged beneath a third) coming together and reacting to form a fourth.

[Nguyen et al. 2002] used the GFM to model fire where the expansion of fuel into products admits jumps in both velocity and pressure.[5] As in the surface tension case, the pressure jump is incorporated directly into the Poisson equation. The velocity jump needs to be handled whenever information is combined from different sides of the interface. They implement this by storing two velocity fields, one for fuel and one for products, that inherently store the jump conditions. While this only doubles their storage, the storage requirements would scale linearly with the number of different materials. We instead compute ghost values on the fly by generalizing the concept of a scalar or vector field to encapsulate the application of the jump condition. Our implementation wraps the data in a *lookup* class that maintains a state variable indicating the region for which values are being looked up. For example, when using semi-Lagrangian advection to update a face velocity in region i, we create a lookup class instance and set its internal state to indicate that any queried values should be returned with respect to region i. Then the lookup class ensures that any data used to construct the ray or interpolate is retrieved with the proper jump conditions already applied. In this manner, existing interpolation and discretization code is generalized with relative ease to account for discontinuities. The lookup classes can also be used to incorporate object intersections in the same manner as jump discontinuities. For example, for thin objects, a nested lookup class can be used to check for object intersections and return the appropriate object ghost velocity as described in [Guendelman et al. 2005], simplifying object interaction implementations significantly.

[5] See equations (2) and (3) for the jump conditions

[Nguyen et al. 2002] used only the normal component of the velocity to advect the fuel level set, which is sufficient for their WENO scheme that was applied without particles. We instead use semi-Lagrangian advection for the level set equation, and this requires the tangential component of velocity as well in order to properly trace characteristics. Since the tangential component is continuous across the interface, we form the normal component as usual and simply add the tangential component from the local fluid velocity. The tangential component is required for particle advection as well.

8 Examples

To demonstrate the effectiveness of our approach, we simulated a number of examples that range in resolution from 150^3 to 350^3 on a number of 4 processor Opteron machines. The computational cost for the examples range from 5 to 50 minutes per frame. Surface tension was the main cause for the examples with slower simulation times. We augmented a standard ray tracer to use the same projection based querying of the level set functions that the simulation uses, since rendering each level set independently can lead to multiple intersections per interface (from numerical error).

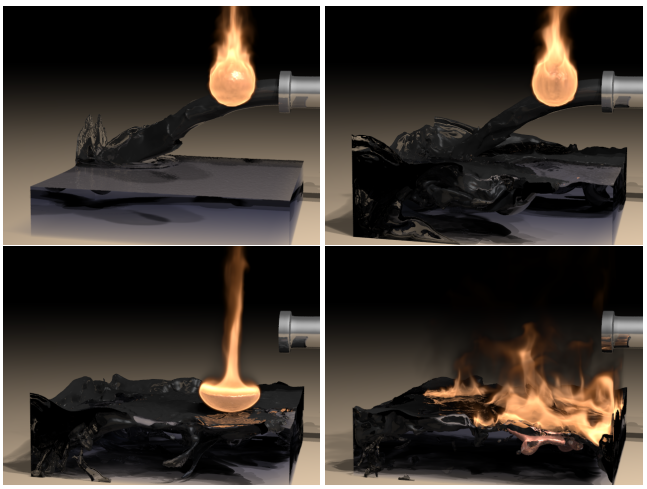


Figure 12: Oil pouring into water, then catching on fire. Note that the fiery ball is a separate phase of fluid, and that it deforms into the shape of a droplet as it falls (200^3 grid, 4 phases).

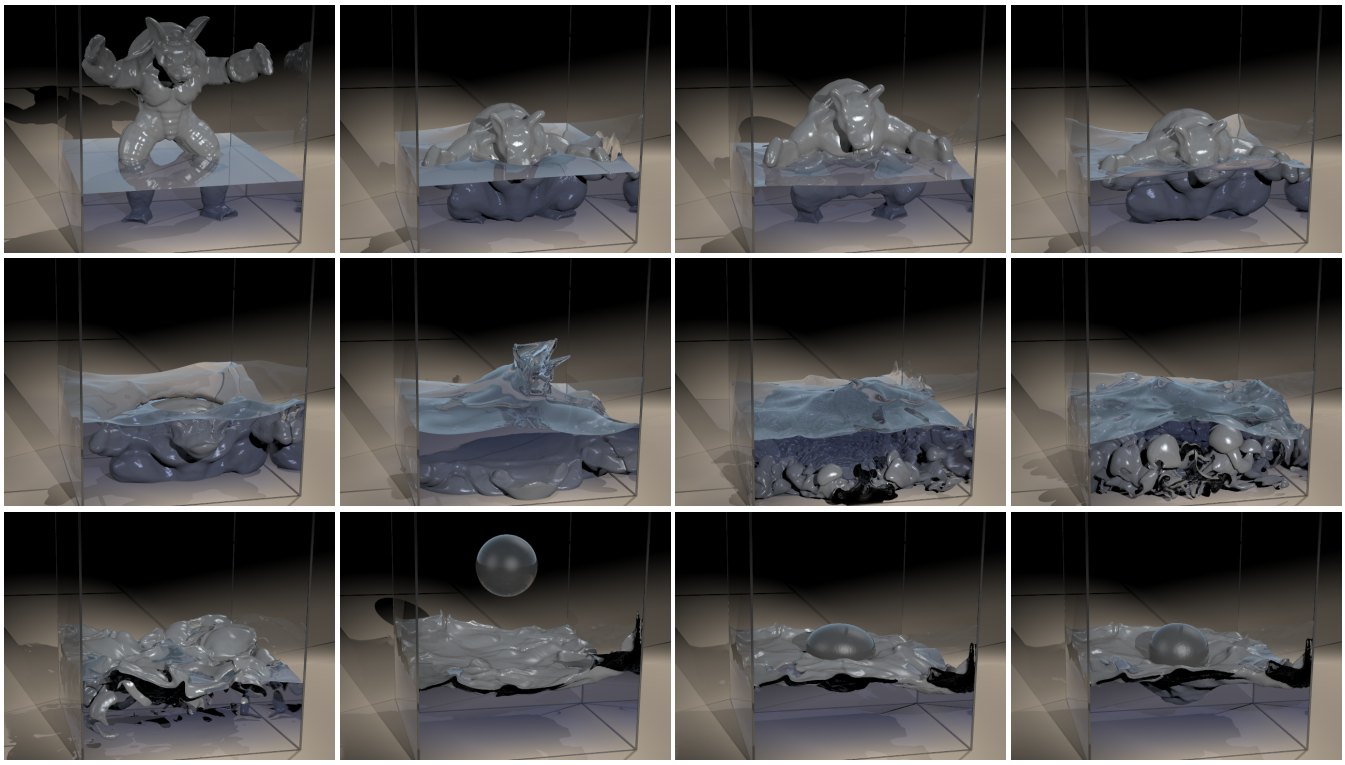


Figure 13: An armadillo that starts out viscoelastic, becomes viscous and more dense than the water, then inviscid and lighter than the water, and finally viscoelastic again before another viscoelastic liquid is dropped onto it ($250 \times 275 \times 250$ grid, 4 phases).

Figure 10 depicts two drops suspended in liquid. The lower drop has a lower density than the surrounding fluid and the upper drop has a higher density. Both drops have surface tension. Figure 3 depicts a kinematic sphere splashing into a number of liquids. The air region is simulated as a Dirichlet region, and the liquids are of increasing density from top to bottom. Figure 6 shows four layers of fluid where the lower middle liquid is lighter than the top middle liquid. This causes a Rayleigh-Taylor instability as the two liquids switch places.

Figure 8 shows a number of different fluids on an inclined plane. The liquid with the highest viscosity is blue, then green, then silver, and finally the clear water is simulated as inviscid. Figure 13 depicts a viscoelastic armadillo in a pool of water. The viscoelastic property is then removed making it viscous only, then the viscosity is turned off and the density is turned down making it bubble up to the surface. The former armadillo is then changed back to viscoelastic and a newly introduced viscoelastic liquid is dropped on top of it.

Figure 12 depicts flammable oil being released into a tank of water. The oil rises to make a layer on the surface, which is then ignited. This example uses a temperature based ignition model where the temperature in actively burning regions is T_{max}, but a lower $T_{ignition}$ is needed to cause ignition of surrounding fluid. Figure 11 shows two viscous liquids that react with each other creating a third (air) that bubbles up to the surface. In Figure 7, eight letters with various high densities and viscosities splash into a pool of water and sink to the bottom. The air is treated as an empty region. In the second part of the simulation, the letters are changed to be low density fuels with surface tension, and the air is fully simulated. The letters then rise through the water, bursting into flames as they break the surface.

9 Conclusions and Future Work

Notably, the new technique does exacerbate the limitations of the original particle level set method with regards to volume loss by facilitating increased scene complexity. This is evident in the Rayleigh-Taylor simulation depicted in Figure 6. In that example, the majority of the mass loss occurs away from triple points, in regions where our method is identical to the original particle level set method (see section 3.1). Of course, this can be addressed by increasing the particle count for the particle level set method or using an adaptive octree or run length encoded type level set simulation, at the expense of increased code complexity and/or CPU time.

10 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, ONR N00014-05-1-0479 (for a SUN computing cluster), ARO DAAD19-03-1-0331, NSF IIS-0326388, NSF ITR-0205671, NSF ITR-0121288, NSF ACI-0323866 and NIH U54-GM072970.

References

CARLSON, M., MUCHA, P., VAN HORN, R., AND TURK, G. 2002. Melting and flowing. In *ACM SIGGRAPH Symp. on Comput. Anim.*, 167–174.

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. *Comput. Graph. and Image Processing 57*, 107–116.

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.

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

FEDKIW, R., ASLAM, T., MERRIMAN, B., AND OSHER, S. 1999. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys. 152*, 457–492.

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

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

FELDMAN, B., O'BRIEN, J., AND KLINGNER, B. 2005. Animating gases with hybrid meshes. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 904–909.

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 *Comput. Graph. Int.*, 178–188.

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

GASCUEL, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. In *Proc. SIGGRAPH 93*, 313–320.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 463–467.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 973–981.

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

HONG, J.-M., AND KIM, C.-H. 2005. Discontinuous fluids. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 915–919.

HONG, J.-M. 2005. *Visual Simulation of Fluids with Discontinuous State Variables*. PhD thesis, Korea University.

HOUSTON, B., NIELSEN, M., BATTY, C., NILSSON, O., AND MUSETH, K. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1, 1–24.

IHM, I., KANG, B., AND CHA, D. 2004. Animation of reactive gaseous fluids through chemical kinetics. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 203–212.

IRVING, G. 2007. PhD thesis, Stanford University.

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 *Comput. Graph. (Proc. of SIGGRAPH 90)*, vol. 24, 49–57.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRÉ, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Eurographics Symp. on Point-Based Graph.*

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

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 457–462.

LOSASSO, F., IRVING, G., GUENDELMAN, E., AND FEDKIW, R. 2006. Melting and burning solids into liquids and gases. *IEEE Trans. on Vis. and Comput. Graph. 12*, 3, 343–352.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., AND STAM, J. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 449–456.

MELEK, Z., AND KEYSER, J. 2005. Multi-representation interaction for physically based modeling. In *ACM Symp. on Solid and Physical Modeling*, 187–196.

MERRIMAN, B., BENCE, J., AND OSHER, S. 1994. Motion of multiple junctions: A level set approach. *J. Comput. Phys. 112*, 334–363.

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.

MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. 2005. Particle-based fluid-fluid interaction. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 237–244.

NEFF, M., AND FIUME, E. 1999. A visual model for blast waves and fracture. In *Proc. of Graph. Interface 1999*, 193–202.

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

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.

RUUTH, S. 1998. A diffusion-generated approach to multiphase motion. *J. Comput. Phys. 145*, 166–192.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 910–914.

SHI, L., AND YU, Y. 2005. Taming liquids for rapidly changing targets. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 229–236.

SMITH, K., SOLIS, F., AND CHOPP, D. 2002. A projection method for motion of triple junctions by level sets. *Interfaces and Free Boundaries 4*, 3, 263–276.

STAM, J., AND FIUME, E. 1995. Depicting Fire and Other Gaseous Phenomena Using Diffusion Process. In *Proc. of SIGGRAPH 1995*, 129–136.

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.

VESE, L., AND CHAN, T. 2002. A multiphase level set framework for image segmentation using the mumford and shah model. *Int. J. of Comput. Vision 50*, 3, 271–293.

WANG, H., MUCHA, P., AND TURK, G. 2005. Water drops on surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 921–929.

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

ZHAO, H.-K., CHAN, T., MERRIMAN, B., AND OSHER, S. 1996. A variational level set approach to multiphase motion. *J. Comput. Phys. 127*, 179–195.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. (SIGGRAPH Proc.) 24*, 3, 965–971.