

# Animation and Rendering of Complex Water Surfaces

Douglas Enright  
Stanford University  
Industrial Light & Magic  
enright@stanford.edu

Stephen Marschner  
Stanford University  
srm@graphics.stanford.edu

Ronald Fedkiw  
Stanford University  
Industrial Light & Magic  
fedkiw@cs.stanford.edu

2002/11/18

GFX Lunch

# Summary

This paper is a continuation of previous work, and discusses the methods required to:

- Compute the motion of a fluid
- Smoothly update the position of the liquid surface
- Render the fluid surface

*Main Focus:* Water poses a more difficult problem than smoke because of the discontinuous interface between the water and the air.

# Contents

## 1. Summary

## 2. Background

- (a) Surface tracking (Level set methods)
- (b) Rendering (Photon mapping)
- (c) Fluid simulation (Governing equations, discretization, methods)

## 3. New work

- (a) Particle level set method
- (b) Velocity extrapolation
- (c) Results

## 4. Conclusion

## Background

- Surface tracking
  - Osher and Sethian (1988) Level set method (pure Eulerian)
- Rendering
  - Wann-Jensen (1995) Photon mapping
- Fluid simulation
  - Basic computational fluid dynamics
  - Harlow and Welch (1965) Marker-and-Cell technique for grid calculations
  - Kass and Miller (1990) Heightfield solution to wave equation
  - Stam (1999) Stable (implicit) fluid dynamics in three dimensions
  - Foster and Fedkiw (2001) Refined surface description for liquids

## Osher and Sethian (1988)

Introduce level set method, a convenient way of defining multiple fluids in an Eulerian (grid-only) sense.

The water surface is defined by  $\phi = 0$  isosurface of the field value  $\phi$ . Foster and Fedkiw (2001) represent the inside the liquid volume as  $\phi \leq 0$ , and outside as  $\phi > 0$ .

The advection equation for  $\phi$  is

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \right) \phi = 0 \quad (1)$$

The normal vector can be easily calculated from  $\phi$ :  $\hat{n} = \nabla\phi/|\nabla\phi|$

## Wann-Jensen (1995)

Developed *photon mapping*, a Monte-Carlo method for global illumination and interfacial light transport; this technique captures not only the indirect illumination in a scene (radiosity-like), but also the light caustics caused by refraction of light through a moving density interface.

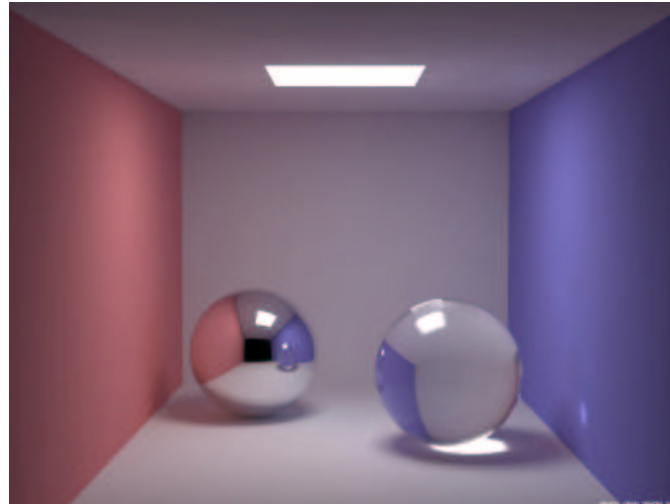


Figure 1: Image rendered using photon mapping, note global illumination and caustics

Photon mapping works something like this:

1. *Population phase*: Photons are traced from light sources into the scene; they react with reflective and refractive surfaces and will eventually be absorbed by a surface—this location and the photon's energy are stored.
2. *Collection phase*: The scene is raytraced using standard methods, but when the ray reaches a surface where a light intensity value is desired, a complex averaging procedure determines the proper surface color and intensity.



Figure 2: Caustics from a glass of cognac, rendered using photon mapping with 200,000 photons; smooth caustics require a smooth refraction surface and sufficient photon density

## Equations of fluid motion

Assuming density ( $\rho$ ) and temperature are nearly constant, a flow can be described by the Navier-Stokes equations, written here in pressure-velocity variables:

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

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla - \nu \nabla^2 \right) \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p + \mathbf{f} \quad (3)$$

- where  $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$  is the gradient operator in 3-D,
- $(\partial/\partial t + \mathbf{u} \cdot \nabla - \nu \nabla^2)$  is the frequently-recurring convection-diffusion operator, and
- $\mathbf{f}$  is an external force per unit mass.

## Explicit vs. implicit formulations

In order to solve these systems, the volumes are discretized into cells of size  $\Delta x$ , and the equations above are converted into discrete equations. Both spatial and temporal derivatives are rewritten.

A second-order explicit formulation for the first derivative is:

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2) \quad (4)$$

And a similar-order implicit formulation looks like:

$$\frac{1}{2} \left[ \left(\frac{\partial u}{\partial x}\right)_i + \left(\frac{\partial u}{\partial x}\right)_{i+1} \right] = \frac{u_{i+1} - u_i}{\Delta x} + \mathcal{O}(\Delta x^2) \quad (5)$$

Advantages and disadvantages of each method include:

- Explicit
  - unknown quantity solved using *only* known quantities
  - requires wider “footprint” for given order of accuracy
  - usually requires small time step to be stable
  
- Implicit
  - solving for unknown quantities requires matrix solution
  - requires smaller “footprint” for given order of accuracy
  - can accept much larger time step and remain stable

## Harlow and Welch (1965)

Established the now-popular MAC (Marker And Cell) method for grid-based (Eulerian) computational fluid dynamics

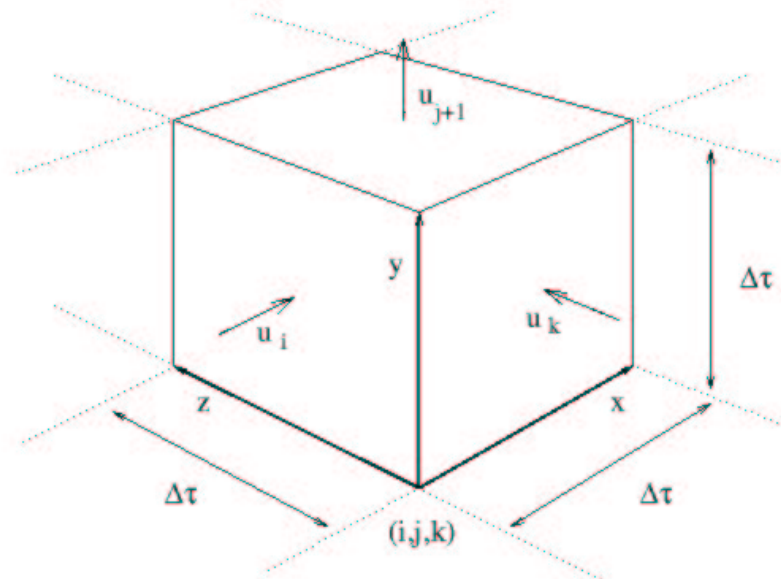


Figure 3: Classic marker-and-cell staggered grid cell

On this grid, solve the 2D incompressible Navier-Stokes equations in pressure-velocity:

$$\nabla \cdot \mathbf{u} = 0 \quad (6)$$

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla - \nu \nabla^2 \right) \mathbf{u} = \mathbf{g} - \frac{1}{\rho} \nabla p \quad (7)$$

The spatial and temporal discretizations are explicit.

*Marker particles* placed on the surface are advected at the local velocity and determine the configuration of the free surface.

Velocities normal to walls are set to zero, and the pressure in the interface cells is set to atmospheric.

## Kass and Miller (1990)

Created a stable (implicit) fluid solver for 1-D and 2-D shallow-water equations (wave equation).

With  $h(x, y)$  as the height of the water surface, and  $b(x, y)$  the height of the ground, set  $d(x, y) = h - b$  as the depth of the water; we write the equations for conservation of mass and momentum (this time using velocity and depth variables) as:

$$\frac{\partial d}{\partial t} + \frac{\partial ud}{\partial x} = 0 \quad (8)$$

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \right) \mathbf{u} = -g \frac{\partial h}{\partial x} \quad (9)$$

If one ignores the non-linear advection term  $(\mathbf{u} \cdot \nabla \mathbf{u})$ , one can differentiate and rearrange the terms to write the equation:

$$\frac{\partial^2 h}{\partial t^2} = gd \nabla^2 h \quad (10)$$

which is the wave equation with wave speed  $\sqrt{gd}$ .

In 1-D the implicit discretization creates a tridiagonal matrix for  $h(x)$ , and in 2-D, the wave equation solution is split and solved one dimension at a time (iterating until a stable solution is reached).

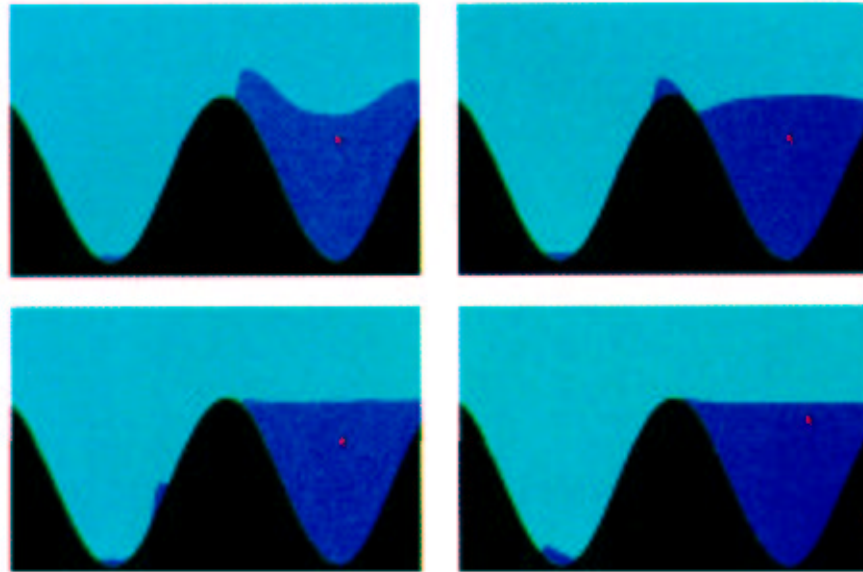


Figure 4: 1-D dynamic heightfield simulation results

This paper also presents results for 2-D heightfields. Later, Foster and Metaxas (1996) reworked the 2-D heightfield case, but ejected particles to simulate splashing, and did not throw away the non-linear advection term.

## Stam (1999)

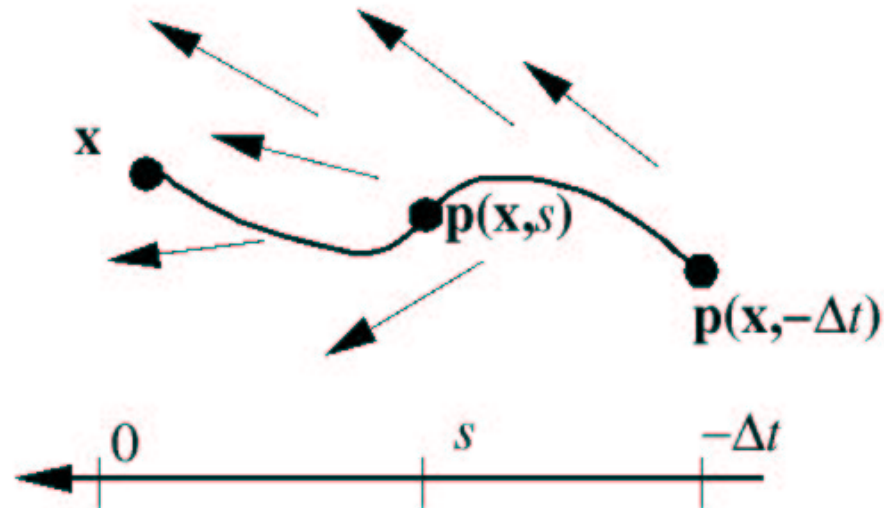
- A true 3-D solver, unlike heightfields used by Kass and Miller (1990)
- Uses a semi-Lagrangian scheme for advection first proposed by Courant, Isaacson, and Rees (1952)
- Usable only for gaseous-like phenomena as advection method causes significant dissipation
- 2-D version runs in real time (interactive), 3-D in near-real time at low enough resolutions

The procedure is as follows:

1. Start with  $\mathbf{u}(\mathbf{x}, t)$  and  $\Delta t$ ,
2. Modify it for external forces ( $\mathbf{f}$ ) using explicit step,
3. Modify it for advection using semi-Lagrangian formulation,
4. Modify it for diffusion using implicit solution to Poisson equation,
5. Project the resulting divergent velocity field onto a divergence-free field, call it  $\mathbf{u}(\mathbf{x}, t + \Delta t)$

## Stam: 3. Advection step

The new velocity at point  $x$  is the velocity at a point traced backwards  $\Delta t$  on a frozen streamline through point  $x$ .



This method is unconditionally stable but unrecoverably diffusive.

## Stam: 4. Diffusion step

To allow the user tunable diffusion, a diffusion step is included. This step uses an implicit formulation for diffusion because the explicit formulation is unstable when viscosity is large (though Foster and Fedkiw, 2001, use the explicit version).

Using the author's notation:

$$(\mathbf{I} - \nu \Delta t \nabla^2) \mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x}) \quad (11)$$

This step requires solution of a sparse linear system, which the author claims can be done in linear time:  $\mathcal{O}(M)$ .

## Stam: 5. Projection step

The previous steps produce a divergent velocity field ( $\nabla \cdot \mathbf{u} \neq 0$ ).

Projecting this velocity field onto a divergence-free velocity field requires use of the *Helmholtz-Hodge Decomposition*, which states that a vector field ( $\mathbf{w}$ ) can be written as the sum of a divergence-free vector field ( $\mathbf{u}$ ) and the gradient of a scalar-valued dilatation field ( $q$ ):

$$\mathbf{w} = \mathbf{u} + \nabla q \quad (12)$$

The divergence of this equation is:

$$\nabla \cdot \mathbf{w} = \nabla^2 q \quad (13)$$

which is a Poisson equation for  $q$ , and another sparse linear system.

## Stam: Additional calculations

Smoke densities were spread using the convection-diffusion operator with source and sink terms.

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla - \kappa_a \nabla^2 \right) a = S_a - \alpha_a a \quad (14)$$

These fields were rendered in hardware as an overlapping series of transparency layers.

## Stam: Results

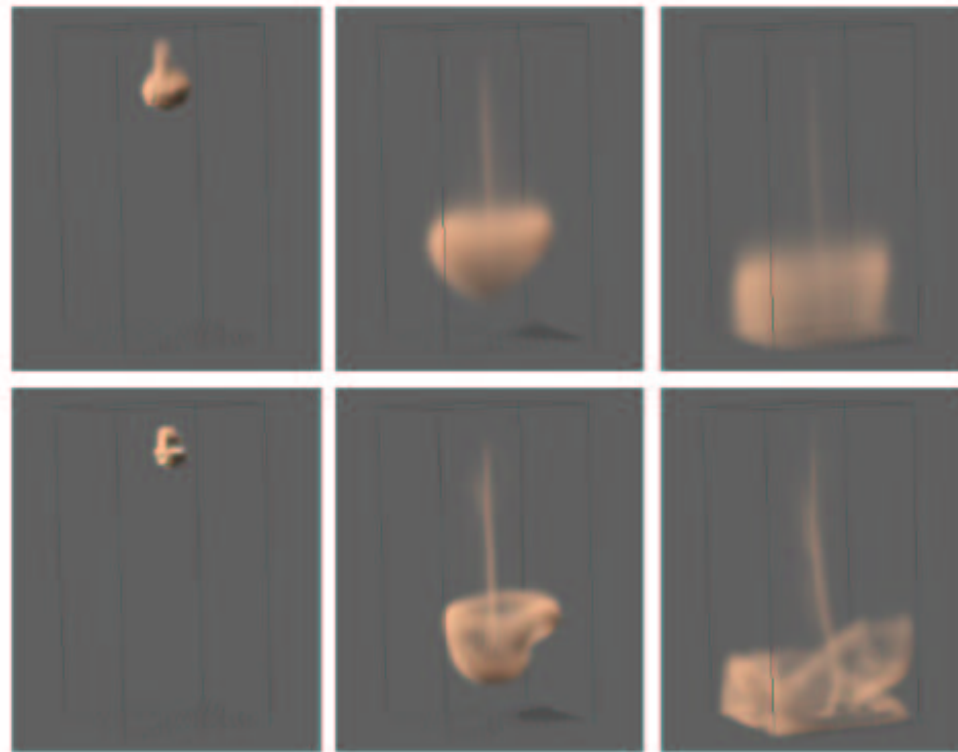


Figure 5: Comparison of 3-D smoke tracking using linear (Stam 1999) vs. cubic (Fedkiw, Stam, Jensen 2001) interpolation

## Foster and Fedkiw (2001)

The methods in this paper lay the foundation of the majority of the work done in Enright, Marschner and Fedkiw (2002), including:

- Fluid solver from Stam (1999), but
  - Use explicit diffusion step
  - Enforce divergence-free velocity field only after accounting for solid objects
- Combined level set and Lagrangian front-tracking method for surface definition
- New approach for calculating flow around objects

Their procedure is as follows:

1. Discretize volume of fluid and solids into voxels
2. Model liquid interface with particles and implicit surface
3. Update the grid velocities as per Stam (1999)
4. Apply velocity constraints due to objects (moving or not)
5. Enforce incompressibility by solving Poisson equation
6. Update interface location, surface particles and implicit surface
7. Repeat steps 3-6

## FF: 2. Model liquid surface

Uses a hybrid method that combined pure Lagrangian particle tracking with purely Eulerian level set method.

First, particles are placed to completely fill the inside of the initial fluid volume.

From these particles, an implicit function  $\phi(x)$  is created on a high-resolution grid and smoothed.

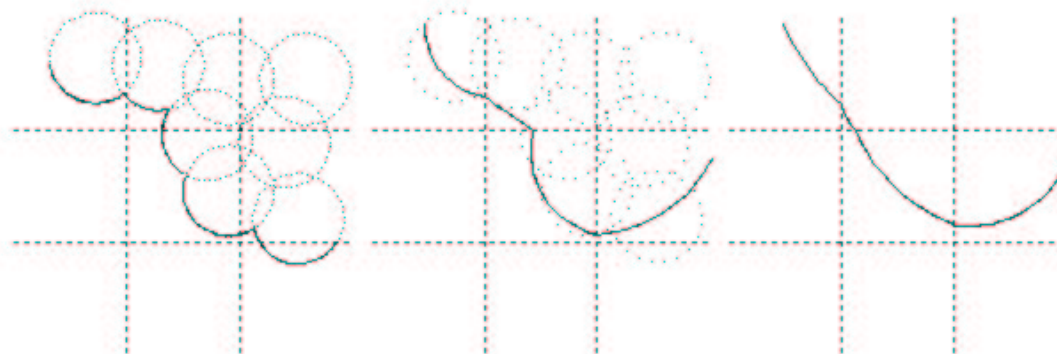


Figure 6: True isosurface of implicit function, interpolated, then smoothed

## FF: 3. Update velocities

The velocity calculation more-or-less follows that of Stam (1999). Boundaries are treated based on the contents of the cell.

- Cells completely in the air phase have their velocities set to zero—assumes that air dynamics have negligible effect (caused problem in Shrek)
- Cells on the liquid-air interface have their pressure set to atmospheric and have their velocities adjusted to explicitly enforce incompressibility.
- Cells inside solid objects have their velocities set to that of the object (see next slide)

## **FF: 4. Apply velocity constraints**

To account for the effects of moving, solid objects in the flow, perform the following steps:

1. During grid velocity calculation, treat any cells inside solid objects as fluid cells with a velocity equal to the given object's velocity, including rotation.
2. Each fluid cell that intersects a solid boundary gets its component normal to the object set to zero
3. Reset the velocities of cells within solid objects to the given object's velocity, including rotation.
4. Continue on to the next step (enforce incompressibility) and hold fixed the velocity for those cells within solid objects.

## FF: 5. Enforce incompressibility

A method almost identical to that of Stam (1999) is used, only instead of an arbitrarily-scaled dilatation field  $q$ , Foster and Fedkiw write a Poisson equation for the pressure  $p$ :

$$\nabla^2 p = \frac{\rho}{\Delta t} \nabla \cdot \mathbf{u} \quad (15)$$

This pressure is then used to update the individual velocities (authors' notation):

$$u_{ijk}^{t+\Delta t} = u_{ijk} - \frac{\Delta t}{\rho \Delta \tau} (p_n - p_{n-1}) \quad (16)$$

## FF: 6. Update surface

The isosurface (level set) defined by  $\phi(\boldsymbol{x})$  is updated using a high-order upwind differencing scheme (on the higher resolution grid, presumably). Osher and Sethian (1988) gave the equation to update the function  $\phi$  as:

$$\left( \frac{\partial}{\partial t} + \boldsymbol{u} \cdot \nabla \right) \phi = 0 \quad (17)$$

The particles making up the surface are advected according to the local velocity, and any particles more than a few grid cells away from the surface are removed.

In areas where the curvature of  $\phi$  is large, the particles will modify the local value of  $\phi$ . This corresponds to areas where splashing occurs.

## FF: Results

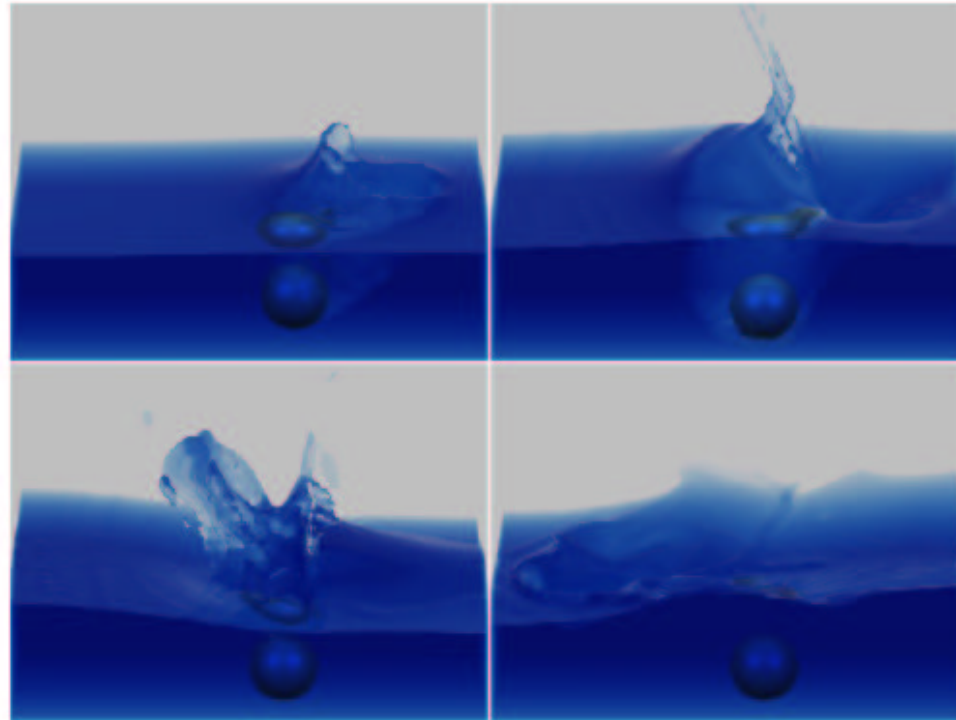


Figure 7: Spherical ball moves into fluid volume, 140x110x90 grid cells

## Enright, Marschner, and Fedkiw (2002)—Finally

EMF build on the work of Foster and Fedkiw (2001), adding:

- A new “thickened” front-tracking technique
- New velocity extrapolation method to allow more control over water surface appearance
- Integration with photon mapping renderer

The method, like others, still uses 1st order Euler time step integration for all advection operations, and 1st order trilinear interpolation for grid-particle or particle-grid operations.

## Particle level set method

- Method is inspired by the feature/detail loss inherent in pure level set methods, and the failure of the one-sided hybrid liquid volume method to model the air phase.
- Represents a shift away from volume methods and to surface methods
- Uses  $\approx 64$  particles per grid cell on both sides of surface to define the surface
- Particle radii set to be tangent to surface whenever possible
- During advection, any “escaped” particles are used to correct the implicit function  $\phi$ .

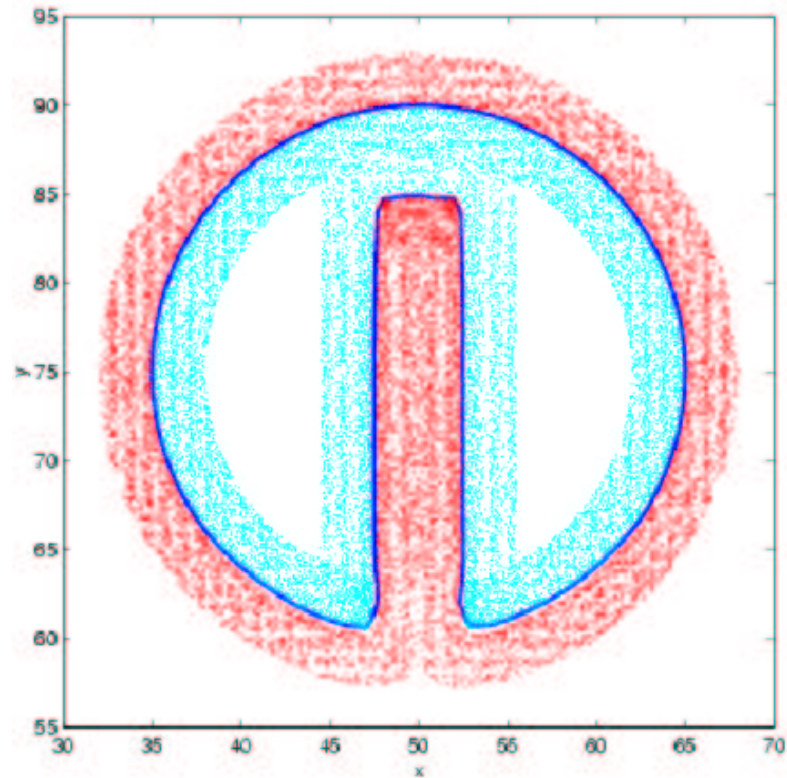


Figure 8: Example surface showing marker particles on both sides of interface

## Velocity extrapolation

The essence of this method is the extrapolation of velocities farther inside the air volume from the air-water interface. This allows the use of higher-order methods for advection of level set and particles.

The method solves for each of the component velocities, one cell at a time, using the equation:

$$\frac{\partial u}{\partial \tau} = -\hat{n} \cdot \nabla u \quad (18)$$

where  $\tau$  is pseudo-time, and  $\hat{n}$  is the normal vector defined by the level set ( $\hat{n} = \nabla \phi / |\nabla \phi|$ )

Because this is purely ad-hoc, this extrapolated velocity can be averaged with a desired wind field to churn up or suppress the surface motion.

## Results

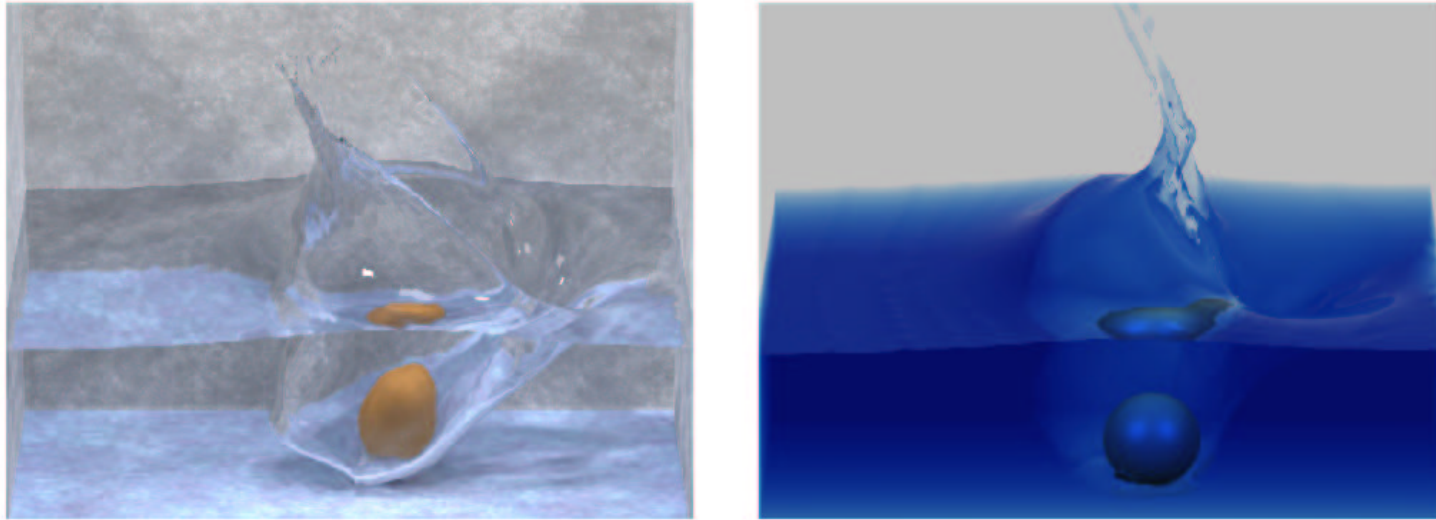


Figure 9: Spherical ball moved into fluid, 140x110x90 grid cells

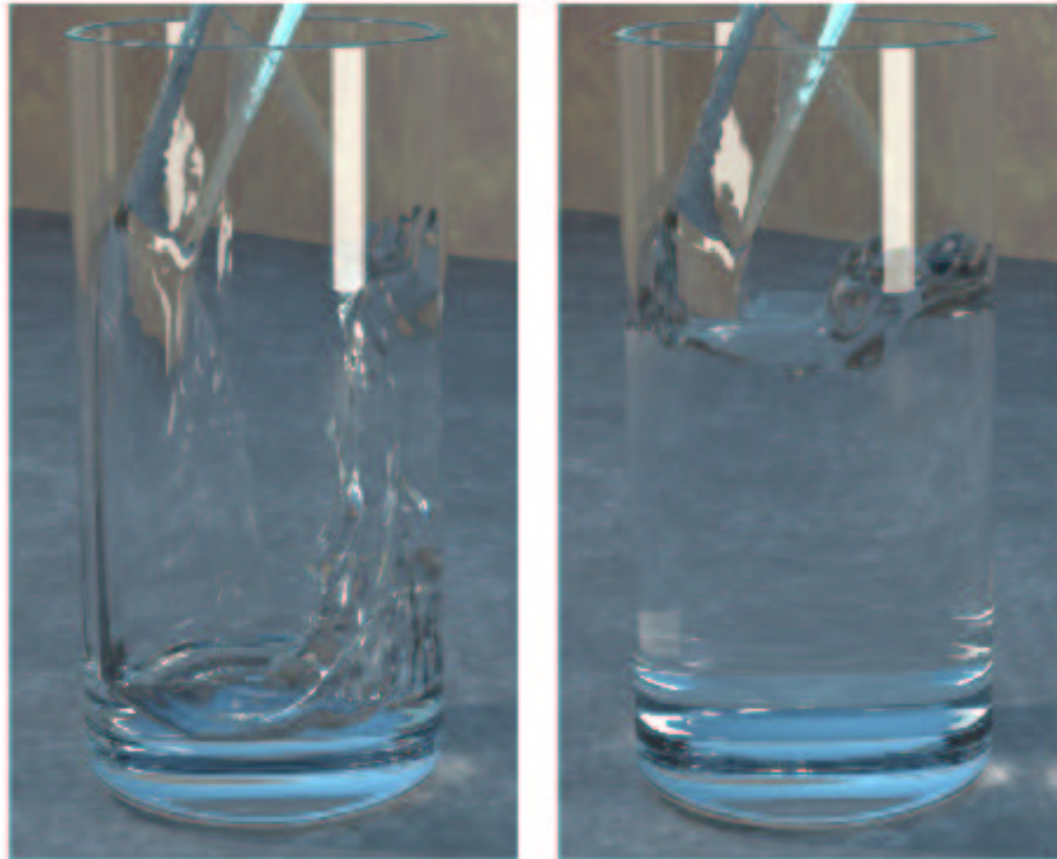


Figure 10: Water being poured into a clear glass, 55x55x120 grid cells

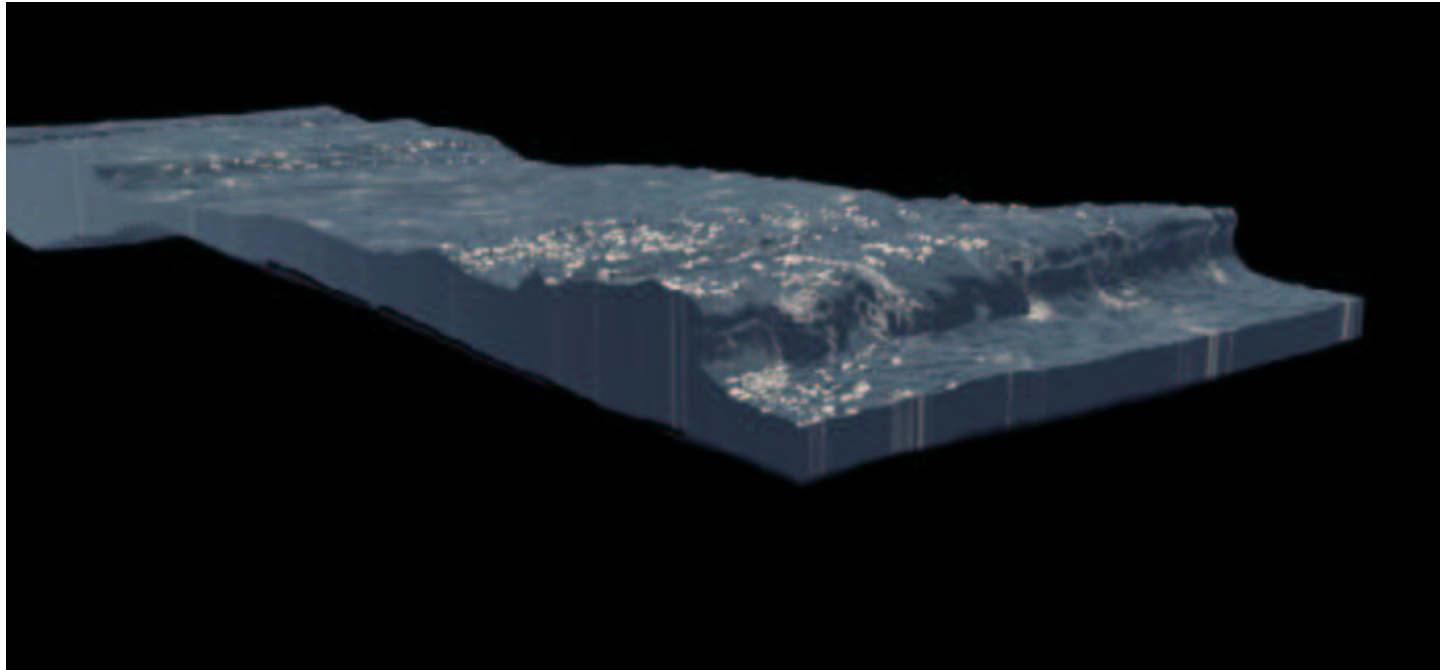


Figure 11: Wave breaking on a submerged shelf, 540x75x120 grid cells, rendered by proprietary software at ILM

## Conclusion

Enright, Marschner, and Fedkiw (2002) have extended the previous work of Foster and Fedkiw (2001), producing an evolutionary improvement in the simulation and rendering of water for a CG production environment.

Its advantages include:

- tunable parameters allow some control over dynamics
- rendering step properly accounts for all light transport in the scene

Its disadvantages include:

- conservation of mass not enforced—causes “popping” of features