Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid

Mark Carlson Peter J. Mucha Greg Turk

Georgia Institute of Technology*

Figure 1: A silver block catapulting some wooden blocks into an oncoming wall of water.

Abstract

We present the *Rigid Fluid* method, a technique for animating the interplay between rigid bodies and viscous incompressible fluid with free surfaces. We use distributed Lagrange multipliers to ensure two-way coupling that generates realistic motion for both the solid objects and the fluid as they interact with one another. We call our method the *rigid fluid* method because the simulator treats the rigid objects as if they were made of fluid. The rigidity of such an object is maintained by identifying the region of the velocity field that is inside the object and constraining those velocities to be rigid body motion. The rigid fluid method is straightforward to implement, incurs very little computational overhead, and can be added as a bridge between current fluid simulators and rigid body solvers. Many solid objects of different densities (*e.g.*, wood or lead) can be combined in the same animation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: physically based animation, rigid bodies, computational fluid dynamics, two-way coupling

1 Introduction

Solid objects interact with fluids every day–our children play with rubber duckies in the tub, athletes dive into swimming pools, and ice clinks in our glass as we pour in our delicious Tang. To simulate these kinds of motion we must first describe the types of interaction, or coupling, the solids and fluid can have [O'Brien et al. 2000]. We distinguish between three types of coupling: one-way solid-to-fluid coupling, one-way fluid-to-solid coupling, and two-way coupling.

It is common in computer animation to see a ball splash into a pool of liquid [Foster and Metaxas 1997; Foster and Fedkiw 2001; Enright et al. 2002b]. This is an example of one-way solid-to-fluid coupling where the motion of the ball is predetermined and the fluid motion is a secondary effect in response to the ball. In such simulations, the fluid has no effect on the motion path of the ball, but the ball can splash the water all around.

In one-way fluid-to-solid coupling, the fluid moves the solid without the solid affecting the fluid. Foster and Metaxas demonstrate this type of coupling by animating tin cans floating on top of swelling water [1996]. In this type of one-way coupling the tin can could shrink to the size of a cork or grow to the size of a barrel without affecting the motion of the water.

With two-way coupling of solids and fluid, simulation alone can drive many scenes that once required assistance from hand animation. For example, flood waters could sweep away a score of horseback riders, washing around them before they can reach the safety of a hastily built wall of stones, the flood water slowing only briefly as it breaks through and washes away the makeshift barrier. Alternatively, a doomed battleship, cracked in half by torpedoes, would list and sink realistically, causing eddies and whirlpools, possibly taking a few unfortunate seamen down with the undertow.

This work focuses on two-way coupling of rigid bodies and incompressible fluid. Two-way coupling of this type is in general a difficult problem [Fedkiw 2002], but with the *rigid fluid* method, two-way coupling between fluid and rigid bodies is a straightforward addition to a fluid and rigid body solver, and the extra computational cost scales linearly with the number of rigid bodies.

By changing the density of a ball in the rigid fluid method, we can achieve vastly differing effects in the ball-splashing-into-liquid animation. If the ball is made of lead it will create a large splash and rapidly sink to the bottom, but if the ball is made of wood it will create a smaller splash, float to the surface of the liquid and bob about a bit (see figures 2 and 3).

The rest of this paper is organized as follows: Section 2 highlights previous work in solid-fluid coupling. Section 3 details the equations of motion for fluids and general solution techniques. Section 4 emphasizes rigidity as the essential difference between the rigid body and fluid domains. Section 5 explains the equations behind the rigid fluid technique, and section 6 gives implementation details on how we solve those equations. Section 7 describes how to advance the computational domain. Section 8 details some animations created with the rigid fluid method. Finally, we conclude the paper with a discussion of possible future work in section 9.

2 Previous Work

This section covers work in the coupling of physically simulated solids and fluid; for an overview of research in only rigid body or

*e-mail:{carlson@gvu, mucha@math, turk@cc}.gatech.edu

only fluid solvers we recommend [Guendelman et al. 2003] and [Enright et al. 2002b] or [Carlson 2004].

To the best of our knowledge, Chen and da Vitoria Lobo [1995] were the first to solve the Navier-Stokes equations in a computer graphics setting. They solved the two-dimensional equations at interactive rates, and added the third dimension with a height field based on the pressure. They demonstrated both kinds of one-way coupling. In addition, they also proposed tuning the velocity and pressure around objects to achieve two-way coupling, although they did not implement this idea. Many researchers since then have demonstrated one-way solid-to-fluid coupling. In [Foster and Metaxas 1996; Foster and Metaxas 1997; Stam 1999; Fedkiw et al. 2001], the rigid bodies were treated as boundary conditions with set velocities. Foster and Fedkiw [2001] improved on that technique by allowing the fluid to move freely along the tangent of the solids. This improvement was also used in [Enright et al. 2002b].

Foster and Metaxas [1996] demonstrate one-way fluid-to-solid coupling where solids are treated as massless particles that move freely on the fluid's surface.

Yngve et al. [2000] demonstrated two-way coupling of breaking objects and compressible fluids in explosions, however their technique does not apply to incompressible fluids like water.

Takahashi et al. [2002] report two-way coupling of buoyant rigid bodies and incompressible fluids using a combined *Volume Of Fluid* and *Cubic Interpolated Propagation* system. Using a regular grid, they identify any cell that is more than half filled with a rigid body as a solid boundary. They set zero Neumann boundary conditions for the pressure at these boundaries to approximate solid-to-fluid coupling. Takahashi et al. [2003] create a variation of this technique for water with splash and foam. They incorporate a rigid body solver with the fluid solver, and achieve solid-to-fluid coupling by setting the velocity of the fluid inside a cell containing a solid to that of the solid. Both techniques achieve fluid-to-solid coupling by modeling forces due to hydrostatic pressure while neglecting the dynamic forces and torques due to the fluid momentum. Dynamic forces and torques from the fluid are imperative in many of the animations presented in this paper.

Génevaux et al. [2003] demonstrate a type of two-way coupling between an incompressible fluid and deformable solids modeled by mass/spring systems, with a communication interface between the two. However their technique does not easily afford the use of the complex shaped non-deformable rigid-bodies we wish to simulate.

A plethora of research on the coupling of solids and fluid exists in the physics and mathematics literature. Fedkiw uses the *Ghost Fluid* method to couple compressible fluids and deformable solids [2002]. Deformable solids have also been successfully treated with the *Immersed Boundary method* [Peskin 2002].

Two-way coupling between fluids and rigid solids is often accomplished in the computational physics community with the *Arbitrary Lagrangian-Eulerian* (ALE) method, introduced in [Hirt et al. 1974]. The ALE method is a finite element technique and suffers from two main drawbacks. First, the computational grid must be remeshed when the elements get too distorted, an often costly procedure. Second, at least two layers of elements are needed in the gap between solids as they approach one another [Singh et al. 2003].

Researchers studying particulate suspension flows have introduced a two-way coupled computation known as the *Distributed Lagrange Multiplier* (DLM) technique [Glowinski et al. 1999]. The DLM method does not suffer from the need to re-mesh. Our research most closely follows the DLM technique of Patankar et al.[2000] and Patankar [2001]. However, our method uses finite differences instead of finite elements, and the rigid bodies in our method are not restricted to spheres. Our formulation also allows for torques because we can apply any force at any point on the rigid body, not just repulsion forces at the center of mass. Our technique also incorporates free surfaces via level sets [Enright et al. 2002a].

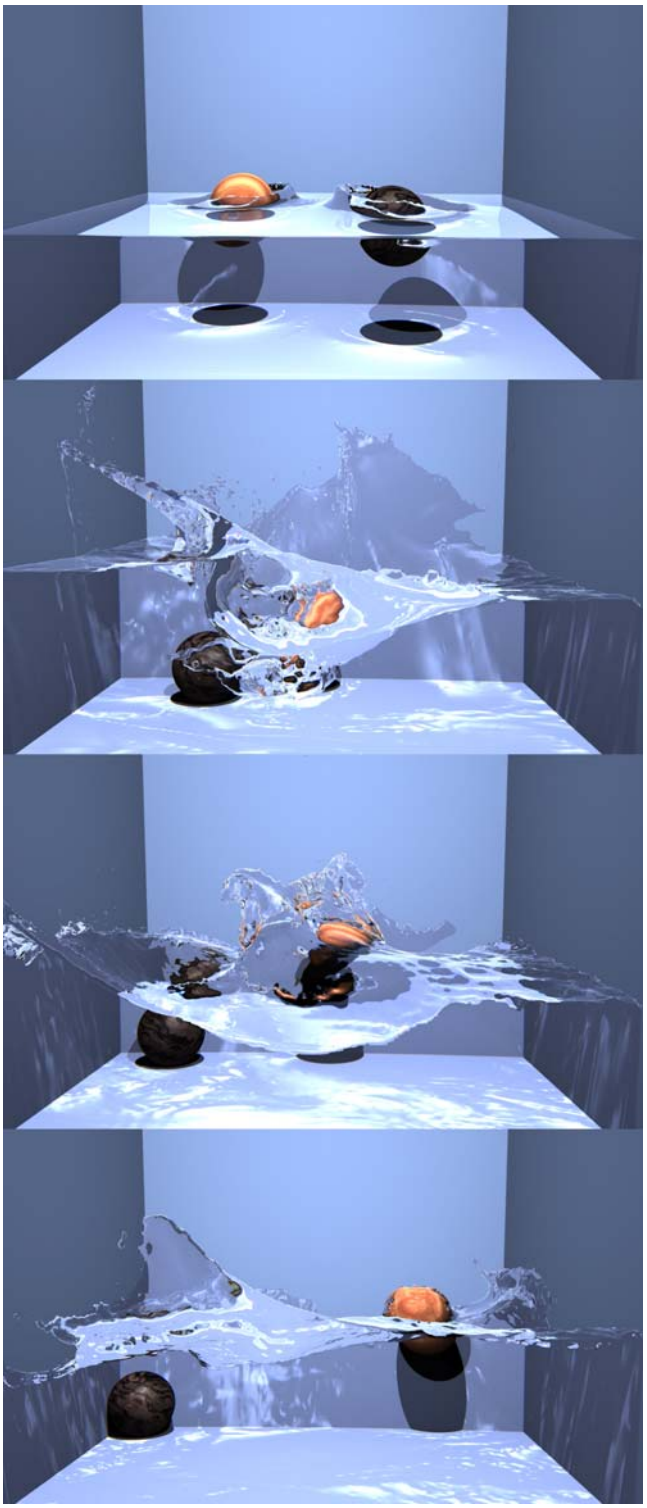

Figure 2: A lead and wood ball are thrown into a tank of water.

3 Equations of Motion for Fluid

The equations of motion for a viscous incompressible fluid are the Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} \; + \; \nabla \cdot (\nu \nabla \mathbf{u}) \; - \; \frac{1}{\rho}\nabla p \; + \; \mathbf{f}. \tag{2}$$

These two equations represent the conservation of mass and momentum, respectively. The vector field $\mathbf{u}_t$ is the time derivative of the fluid velocity. The scalar pressure field is p; ρ is the density of the fluid, and ν is the kinematic viscosity. The vector field $\mathbf{f}$ represents the body force per unit mass; $\mathbf{f}$ is usually just gravity, but it could be the precession of the earth, the wind, or any other user-defined vector field.

A rich history of solving the Navier-Stokes equations exists in computer animation [Chen and da Vitoria Lobo 1995; Foster and Metaxas 1996; Stam 1999; Weimer and Warren 1999; Witting 1999; Fedkiw et al. 2001; Foster and Fedkiw 2001; Carlson et al. 2002; Enright et al. 2002b], but a general overview of the steps that we take to solve them can still be enlightening. There are two major steps to solving for $\mathbf{u}_t$ while enforcing (1), the incompressible fluid constraint.[1]

The first step is to numerically solve for a *best guess velocity*,

$$\tilde{\mathbf{u}} = \mathbf{u} \; + \; \Delta t[\; -(\mathbf{u}\cdot\nabla)\mathbf{u} \; + \; \nabla\cdot(\nu\nabla\mathbf{u}) \; + \; \mathbf{f} \;], \tag{3}$$

without taking into account the pressure. We do not consider the pressure immediately because the current best guess velocity is not divergence free (*i.e.*, $\nabla\cdot\tilde{\mathbf{u}} \neq 0$), and the next step is to use a solution for the pressure to make the new velocity divergence free, thus enforcing the incompressibility constraint.

The second step taken to solve the Navier-Stokes equations is the *pressure projection* step. The term in (2) that we left out of (3) was

$$-\frac{1}{\rho}\nabla p, \tag{4}$$

and we must account for it in the final velocity,

$$\mathbf{u}^{new} = \tilde{\mathbf{u}} \; - \; \frac{\Delta t}{\rho}\nabla p. \tag{5}$$

We also need the final velocity to be incompressible, so we take the divergence of (5) to get

$$\nabla\cdot\mathbf{u}^{new} = \nabla\cdot\tilde{\mathbf{u}} \; - \; \frac{\Delta t}{\rho}\nabla\cdot(\nabla p) = 0. \tag{6}$$

Rearranging (6) gives us the equation

$$\Delta t\nabla^2 p = \rho\nabla\cdot\tilde{\mathbf{u}} \tag{7}$$

with which we must solve for p. We then substitute p back into (5) to complete the pressure projection, thus enforcing the incompressibility constraint. Later we will use a similar projection step to enforce rigidity of solid objects.

4 Rigid Bodies

We have already discussed the Navier-Stokes equations which govern the movement of a viscous incompressible fluid. Before discussing the equations that govern the rigid fluid method we must introduce some notation for the computational domain and describe the *deformation operator*, **D**.

There are two parts to the computational domain, depicted in figure 4. The part of the domain containing only the fluid is $\mathbb{F}$, and the union of cells occupied by the rigid bodies is the solid domain, $\mathbb{R}$. The two domains are disjoint, their shared boundary is $\partial\mathbb{R}$, and together they form the complete computational domain, $\mathbb{C} = \mathbb{F}\cup\mathbb{R}$.

When simulating a rigid body, it is useful to think of its motion as translations and rotations about its center of mass. Simplifying the motion of all points in the rigid body with these assumptions hides the complexity of the rigid body's motion. In essence, rigid body solvers implicity enforce the *rigidity* of the solid by constraining its motion to translations and rotations about the center of mass.

The rigid fluid method, on the other hand, solves the equations of motion for the rigid bodies with equations 1 and 2, so the *rigidity* of the rigid bodies must be explicitly enforced with a Lagrange

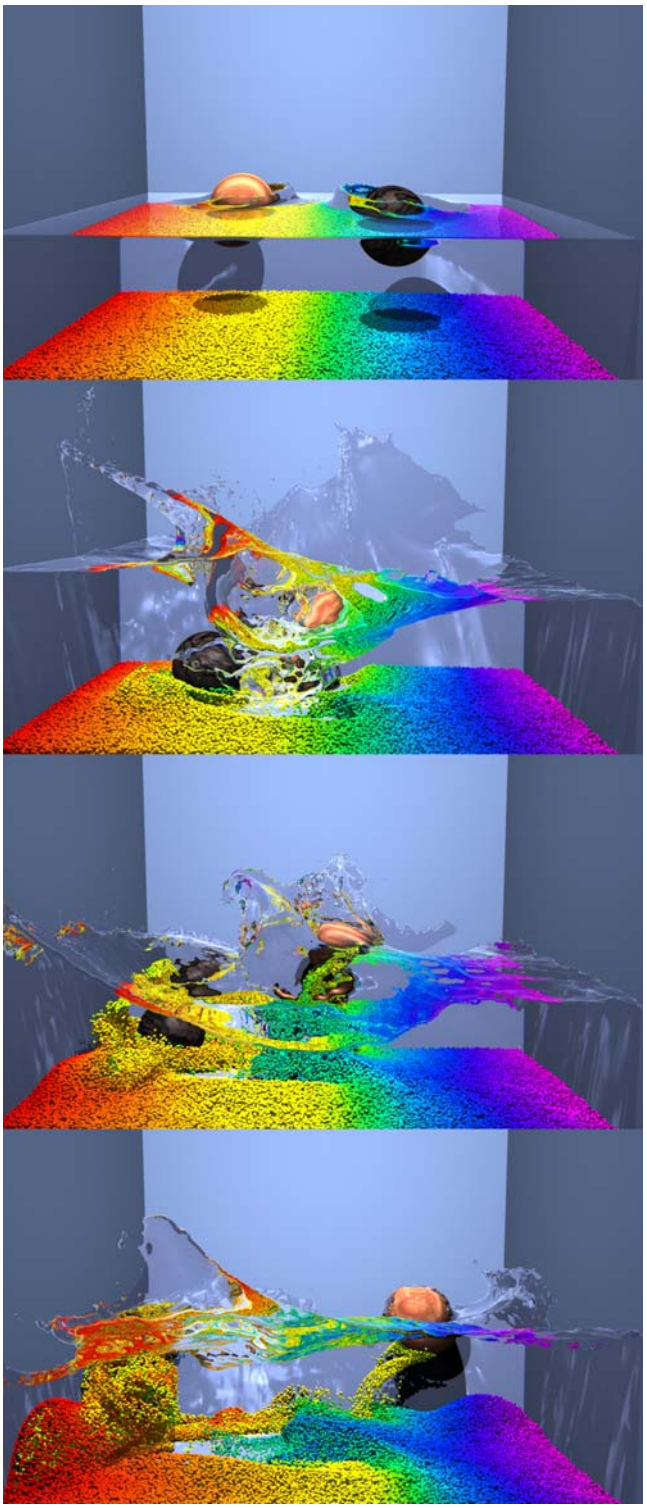

Figure 3: This is the same animation as figure 2, but massless particles are advected to highlight the swirling water motion.

[1] In the following discussion $\mathbf{u}_t \equiv (\mathbf{u}^{new} - \mathbf{u})/\Delta t$, where Δt is the *time step*. This forward Euler formulation is used because it is easy to describe.

Figure 4: The left side of this figure is the compuatational domain, and the right is the rendered frame. On the left the yellow area is the fluid domain $\mathbb{F}$; the blue is the rigid body domain $\mathbb{R}$. Notice that the small blocks on the right are not touching liquid, so they will be controlled by the rigid body solver until they touch liquid.

multiplier. The rigidity constraint enforced on the rigid body domain is very much like the incompressibility constraint discussed in section 3. The rigidity constraint, however, is a stricter constraint, as it is both divergence free, like the incompressibility constraint, and *deformation free*. The rigidity constraint dictates that for every point $\mathbf{y}_j$ in a rigid body, the following relationship must hold:

$$\dot{\mathbf{y}}_j = \mathbf{v} + \omega \times \mathbf{r}_j \tag{8}$$

for some constant $\mathbf{v}$ and ω. In the above, $\dot{\mathbf{y}}_j$ is the velocity at $\mathbf{y}_j$, $\mathbf{r}_j$ is a vector pointing from the rigid body center of mass, $\mathbf{x}$, to $\mathbf{y}_j$, $\mathbf{v}$ is the translational velocity at $\mathbf{x}$, and ω is the rotational velocity about $\mathbf{x}$ along the axis $\omega/|\omega|$ with magnitude $|\omega|$.

The rigidity constraint can be expressed by means of the deformation operator, $\mathbf{D}$, defined for any vector field $\mathbf{u} = (u,v,w)$ by

$$\begin{aligned} \mathbf{D}[\mathbf{u}] &= \frac{1}{2}[\nabla\mathbf{u} + \nabla\mathbf{u}^T] \\ &= \frac{1}{2}\begin{bmatrix} 2u_x & (v_x+u_y) & (w_x+u_z) \\ (v_x+u_y) & 2v_y & (w_y+v_z) \\ (w_x+u_z) & (w_y+v_z) & 2w_z \end{bmatrix}. \end{aligned} \tag{9}$$

The 3×3 symmetric tensor $\mathbf{D}[\mathbf{u}]$ measures the spatial deformation of $\mathbf{u}$. The constraint,

$$\mathbf{D}[\mathbf{u}] = \mathbf{0} \quad \text{in } \mathbb{R}, \tag{10}$$

ensures the motion in $\mathbb{R}$ is in fact rigid body motion [Patankar et al. 2000]. It does not tell us what that motion is, but we know it must agree with equation 8. Depending on the solution procedure employed, it can be advantageous [Glowinski et al. 1999] to use an equivalent differential relationship in place of (10).

5 Governing Equations

In this section we present the governing equations that form the heart of the rigid fluid method. Similar equations were originally derived in [Glowinski et al. 1999] for the DLM method in the weak form appropriate for finite element computations. In contrast, we use a strong form appropriate for finite differences. To streamline our exposition we assume in this section that there is no viscoelastic stress,[2] and all boundary conditions except those on the interface between the rigid bodies and the fluid are understood.

[2] If viscoelastic stress is needed, then the $-\rho_f^{-1}\nabla p$ term in (11) should be changed to $\rho_f^{-1}\nabla\cdot(\Sigma - p\mathbf{1})$ and Σ should be added to the second part of (14). Σ is the extra stress tensor that depends on the deformation rate and deformation history of the fluid at a specific location.

Recalling the definitions of $\mathbb{C}$ and $\mathbf{D}$ from the previous section we are ready to describe the governing equations for the rigid fluid method. The conservation of momentum equations are defined as

$$\mathbf{u}_t = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\cdot(\nu\nabla\mathbf{u}) - \frac{1}{\rho_f}\nabla p + \mathbf{f} \quad \text{in } \mathbb{F} \tag{11}$$

in the fluid domain (same as equation 2), and

$$\mathbf{u}_t = -(\mathbf{u}\cdot\nabla)\mathbf{u} + \quad \nabla\cdot\Pi \quad - \frac{1}{\rho_r}\nabla p + \mathbf{f} \quad \text{in } \mathbb{R} \tag{12}$$

in the solid domain, where ρ_f is the mass density of the fluid, and ρ_r is the rigid body density. The viscous diffusion term is absent in equation 12 because the rigidity constraint already eliminates Newtonian viscous dissipation, however there is an extra term due to the deformation stress inside the solid that is required to maintain rigidity. We implicitly define Π as that extra part of the deformation stress in addition to the harmonic pressure field, p.

Since the deformation-free constraint we enforce on the rigid body domain with equation 10 is stronger than the divergence-free one, we can, for convenience, enforce the divergence-free constraint over the entire domain with the equation

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \mathbb{C}. \tag{13}$$

Constraints (13) and (10) will be enforced by projections onto divergence- and deformation-free motion in the appropriate domains.

The no-slip and dynamic force boundary conditions between the solids and the fluid are defined as

$$\mathbf{u} = \mathbf{u}_i \quad \text{and} \quad (2\rho_f \nu\mathbf{D}[\mathbf{u}] - p\mathbf{1})\cdot\mathbf{n} = \mathbf{t} \quad \text{on } \partial\mathbb{R} \tag{14}$$

where $\mathbf{1}$ is the identity tensor, $\mathbf{u}_i$ and $\mathbf{n}$ are the velocity and normal on $\partial\mathbb{R}$, and $\mathbf{t}$ is the traction force of the fluid on the solid as a sum of the projected viscous stress and pressure. A similar condition can be written for the force of the solid on the fluid in terms of the solid stresses, which must be equal and opposite to $\mathbf{t}$ by Newton's third law. However, we will never need to directly enforce the boundary conditions in (14), as they will be approximately captured by the projection techniques described below.

6 Implementation of Governing Equations

Equations (10)-(14) are the governing equations for all the moving objects in our simulation, both solid and fluid. We solve these equations in three steps. First, we solve (1)-(2) for the entire domain $\mathbb{C} = \mathbb{F}\cup\mathbb{R}$. During this first step, the rigid objects are treated *exactly as if they were fluid*. Next, we calculate the rigid body forces due to collisions and relative density. Finally, we enforce rigid motion for the velocities at those grid locations inside each solid object. These three steps move the simulation forward in time, from $\mathbf{u}^n \to \mathbf{u}^{n+1}$, passing through two successive intermediate stages $\mathbf{u}^*$ and $\hat{\mathbf{u}}$ along the way. In this section we will ignore issues of immobile walls and moving the fluid/air interface, which we will return to in section 7.

6.1 Solving Navier-Stokes Equations: $\mathbf{u}^n \to \mathbf{u}^*$

We first solve the Navier-Stokes equations using an operator splitting scheme [Stam 1999] over all velocities $\mathbf{u}^n$ in $\mathbb{C}$ with four steps:

1. We add the body force, $\Delta t\mathbf{f}$, to all of $\mathbb{C}$. Because rigid body motion will be enforced only inside $\mathbb{R}$, there will be a slip error at $\partial\mathbb{R}$ that increases as $|\rho_r - \rho_f|$ increases. One way to reduce this error, suggested by Patankar [2001], is to add the extra buoyant-weight term, $\Delta t\mathbf{f}(\rho_r - \rho_f)/\rho_f$, to $\mathbb{R}$ at this step. If this is done then $\mathbf{f}$ must be removed from equation 17.

2. We solve the advection term, $-(\mathbf{u}\cdot\nabla)\mathbf{u}$, using the semi-Lagrangian technique which Stam introduced to the graphics community [1999].
3. We solve the diffusion term, $\nabla\cdot(\nu\nabla\mathbf{u})$, using the implicit variable viscosity formulation in [Carlson et al. 2002]; however, we corrected the Dirichlet boundary conditions at the free surface so that we do not cause the velocity dissipation discussed in that paper.
4. We use pressure projection (section 3) to make the velocity in $\mathbb{C}$ divergence free. Because the semi-Lagrangian technique behaves better on a divergence-free velocity field, we have the option in our code to use pressure projection before the advection step in addition to here. All the animations in this paper use this option.

Each of the above steps is stable for large time steps, even with stiff viscous effects. Upon completion of these steps we have the divergence free velocity field $\mathbf{u}^*$ in $\mathbb{C}$, but it is not the final velocity field because we have not accounted for collision and relative density forces of the rigid bodies, nor has the velocity in $\mathbb{R}$ been constrained to rigid body motion.

6.2 Calculating Rigid Body Forces: $\mathbf{u}^* \rightarrow \hat{\mathbf{u}}$

During the time step, the rigid body solver applies collision forces to the solid objects as it updates their positions. These forces must be included in the velocity field to properly transfer momentum between the solid and fluid domains.

As each collision force, $\mathbf{F}_j$, is applied to one of the N rigid bodies, we keep a running sum of the accelerations created on that body over the time step and store it as

$$\mathbf{A}_c = \sum_j \frac{\mathbf{F}_j}{M_i}, \tag{15}$$

where $i \in \{1,2,\ldots,N\}$, and M_i is the mass of the rigid body to which the force is applied.

Similarly, as each force is applied at point $\mathbf{p}_j$, we sum the angular accelerations it creates about each body's center of mass,

$$\alpha_c = \sum_j \mathbf{I}_i^{-1}[\ (\mathbf{p}_j - \mathbf{x}_i)\times\mathbf{F}_j\ ], \tag{16}$$

where $\mathbf{I}_i$ is the moment or inertia of the i^{th} rigid body, in its current orientation, and $\mathbf{x}_i$ is its center of mass.

Forces that arise from the *relative density*, also known as the specific gravity, must also be considered. The relative density of a solid is the ratio of its density to that of the surrounding fluid, ρ_r/ρ_f. If the relative density is greater than 1, then the solid will sink. Conversely, if the relative density is less than 1 the solid will rise and float. It becomes more difficult for the fluid to move an object as the relative density increases. The relative density and collision forces are accounted for in $\mathbb{R}$ with a source term

$$\mathbf{S} = \rho_r\mathbf{A}_c + \mathbf{r}_i\times\rho_r\alpha_c - (\rho_r-\rho_f)[\frac{\mathbf{u}^*-\mathbf{u}^n}{\Delta t} + (\mathbf{u}^*\cdot\nabla)\mathbf{u}^* - \mathbf{f}\ ], \tag{17}$$

where the vectors $\mathbf{r}_i = \mathbf{y}_i - \mathbf{x}_i$ point from the center of mass of the i^{th} rigid body to the grid point locations, $\mathbf{y}_i$, in that rigid bodies domain, $\mathbb{R}_i$. The solution to (17) is direct because all the variables on the right hand side are known.

The relative density term in equation 17 is due to Patankar [2001]. Since he restricts his attention to spherical objects with repulsion forces acting only at the center of mass, Patankar includes an $\mathbf{A}_c$, but not an α_c, collision term. The angular collision terms α_c are essential for the proper treatment of non-spherical objects with collision forces that generate torques.

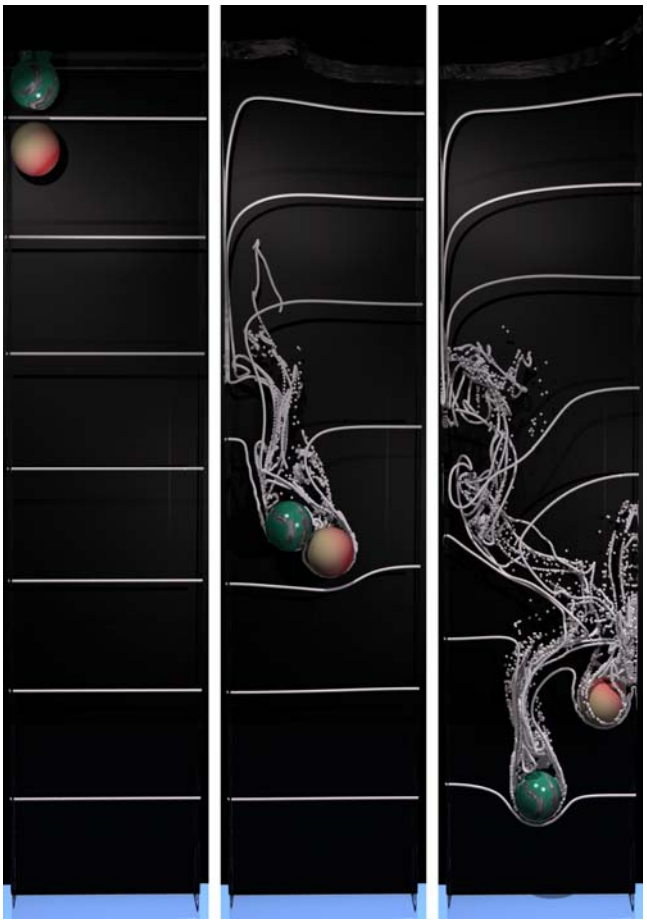

Figure 5: Tumbling of two sinking stones. Passively advected particles mark the fluid movement.

Using $\mathbf{S}$ we solve for a new velocity field,

$$\hat{\mathbf{u}} = \mathbf{u}^* + \ w\frac{\Delta t}{\rho_r}\mathbf{S}, \tag{18}$$

where w is a number between 0 and 1 representing the fraction of volume of a computational cell occupied by the solid. However, $\hat{\mathbf{u}}$ is still not rigid body motion, so we must complete one last step.

6.3 Enforcing Rigid Motion: $\hat{\mathbf{u}} \rightarrow \mathbf{u}^{n+1}$

To guarantee rigid body motion in $\mathbb{R}$, the unknown force, $\mathbf{R}$, that maintains rigidity must be found. Once found, the final velocity,

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}} + \ \frac{\Delta t}{\rho_r}\mathbf{R}, \tag{19}$$

can be solved for in $\mathbb{R}$, but we still do not have an equation for $\mathbf{R}$.

Equation 19 is a projection that enforces the rigidity constraint in much the same way the pressure projection, equation 5, enforces the divergence-free constraint. The constraint enforced by the pressure projection was $\nabla\cdot\mathbf{u} = 0$, and the Lagrange multiplier used to enforce that constraint was p. So, to find an equation for p we took the divergence of (5) and arrived at (7). The constraint that must be enforced with the rigidity projection is equation 10, and the Lagrange multiplier is $\mathbf{R}$, so substituting (19) into (10) yields an equation for $\mathbf{R}$:

$$\mathbf{D}[\mathbf{u}^{n+1}] = \mathbf{D}[\hat{\mathbf{u}} + \frac{\Delta t}{\rho_r}\mathbf{R}] = 0 \tag{20}$$

which states $\hat{\mathbf{u}} + \Delta t \mathbf{R}/\rho_r$ is the desired rigid body motion. Alternatively, we break $\hat{\mathbf{u}}$ in $\mathbb{R}$ into two parts:

$$\hat{\mathbf{u}} = \hat{\mathbf{u}}_R + \hat{\mathbf{u}}', \tag{21}$$

where $\hat{\mathbf{u}}_R$ is the rigid body velocity we are searching for, and

$$\hat{\mathbf{u}}' = -\frac{\Delta t}{\rho_r}\mathbf{R} \tag{22}$$

is due to the stress inside $\mathbb{R}$ that enforces rigid body motion upon it.

As observed by Patankar [2001], the desired rigid body solution of (20) and (22) for $\mathbf{R}$ and $\hat{\mathbf{u}}'$ must conserve momentum and can therefore be obtained directly. Writing (8) as a union over each rigid body yields the equation

$$\hat{\mathbf{u}}_R = \bigcup_i (\hat{\mathbf{v}}_i + \hat{\omega}_i \times \mathbf{r}_i) \tag{23}$$

for some $\hat{\mathbf{v}}_i$ and $\hat{\omega}_i$. Because momentum must be conserved, we obtain $\hat{\mathbf{v}}_i$ and $\hat{\omega}_i$ for each rigid body by directly integrating the intermediate $\hat{\mathbf{u}}$ inside a given rigid body $\mathbb{R}_i$ with the equations:

$$M_i \hat{\mathbf{v}}_i = \int_{\mathbb{R}_i} \rho_i \hat{\mathbf{u}}\, dg_i, \qquad \text{and} \tag{24}$$

$$\mathbf{I}_i \hat{\omega}_i = \int_{\mathbb{R}_i} \mathbf{r}_i \times \rho_i \hat{\mathbf{u}}\, dg_i, \tag{25}$$

where M_i, $\mathbf{I}_i$ and ρ_i are the mass, moment of inertia and density of the i^{th} rigid body, and dg_i is the volume of the grid cell occupied by the solid. Equations 24 and 25 are evaluated by summing the appropriate terms for each grid cell that is fully or partially inside the i^{th} rigid body domain $\mathbb{R}_i$.

Because (19) must be a momentum conserving projection, we simply use (24) and (25) directly to solve for the rigid body velocity $\hat{\mathbf{u}}_R$. We then distribute this rigid body velocity over the objects to get our final velocity:

$$\mathbf{u}^{n+1} = (1-w)\hat{\mathbf{u}} + w\hat{\mathbf{u}}_R, \tag{26}$$

which enforces rigidity and conserves momentum inside $\mathbb{R}$.

7 Advancing the Computational Domain

We solve the rigid fluid equations on a regularly spaced discrete computational domain where the components of the velocity vector are on the faces of the grid cells, and the pressure is in the center. The fluid and solid domains are advanced each time step, so before we solve the rigid fluid equations we determine the new computational domain and identify the grid cells in $\mathbb{F}$ and the grid cells in $\mathbb{R}$. We must also identify the space that $\mathbb{F}$ and $\mathbb{R}$ can not occupy–the immobile boundaries. A static signed distance function, similar to the one used in [Wrenninge 2003], delineates the immobile boundary from the rest of the computational domain. The immobile boundary region has a velocity, so it can be used for objects with one-way solid-to-fluid coupling if the user desires.

To advance the computational domain, $\mathbb{C}$, we must advance both $\mathbb{F}$ and $\mathbb{R}$. We will discuss advancing the fluid domain first.

We use a particle level set, ϕ, to identify the fluid region [Enright et al. 2002a]. In our level set implementation, all grid cells have a width of 1 and the time step is assumed to be 1 for simplicity, but the grid cells in $\mathbb{C}$ have a width of h and will be advanced by Δt. Also, the level set exists on a regular grid, while the velocity in $\mathbb{C}$ is solved on a *staggered grid* [Foster and Metaxas 1996]. Therefore, we compute the velocity for the level set by averaging the velocities at the faces of the cell and then scaling that average by $\Delta t/h$. This not only simplifies the level set implementation, it decouples the time step restrictions of the level set equation from the rest of the simulation. To realistically advance the level set we grow an extension velocity into the regions not filled by the averaging (see [Osher and Fedkiw 2003] for details on extension velocities and moving level sets with external velocity fields).

Figure 6: Two odd shaped gems tumbling in a water filled shaft. Passively advected particles mark the fluid movement.

Once the level set position is updated we identify the new $\mathbb{R}$ by moving the rigid bodies with the solver described in [Guendelman et al. 2003]. The rigid bodies we consider are polygonal objects, possibly concave, and we compute their mass properties from the polygonal representation. The velocities obtained from equations 24 and 25 are used as initial conditions to the rigid body solver. During one rigid body time step, forces and torques are applied to the rigid bodies if there are collisions. As the collision forces are applied, we keep a running sum of the accelerations and angular accelerations they create (equations 15 and 16) and store them in $\mathbf{A}_c$ and α_c. The variables $\mathbf{A}_c$ and α_c represent the accelerations of the rigid bodies due to collisions and are used in equation 17.

After a new position is found for each of the N rigid bodies, we save a list of the grid points that are inside each solid. Just as in [Guendelman et al. 2003], we use a signed distance function for each of our rigid bodies. This function is important because it affords constant time inside/outside tests of the rigid bodies. The speed of this test is important because large objects can take up many of the grid cells. Once we find $\mathbb{R}$ and the list of grid points that reside within it, the fluid domain, $\mathbb{F}$, is found as any grid point not in $\mathbb{R}$ and with $\phi \leq 1/2$.

8 Results

In this section, we describe animations that were created using the rigid fluid technique. The accompanying video includes the full animation sequences.

Figure 1 shows frames from an animation where a silver block,

measuring 20cm in each dimension and with a relative density of 9 is dropped from the top of a one meter tall room. It strikes a plank or wood (relative density 0.74) and catapults several smaller wooden blocks into an oncoming wall of water. The turquoise block's relative density is only two so it slides around more than the silver block. One thing to notice is how the small blocks do cause splashes when they land in the water, but the water also pushes them around so they move with the swells of the sloshing water.

In figures 2 and 3 a lead and a wood sphere (relative densities 11 and 0.55) are thrown into a meter wide tank of water. They have the same initial downward velocity of 3.1m/s, and equal but opposite horizontal velocities of 3.8m/s. They also have rotations of 1000rpms about the vertical axis (looking down on them from above, the lead sphere is spinning clockwise and the wood one is spinning counterclockwise). As revealed by their shadows, the spheres are slightly off center from one another at the start of the simulation but will obviously strike one another. The lead sphere is heavy enough not to be moved much by the wood sphere, and its angular momentum rolls it into the back left corner of the tank as expected, but the lead sphere strikes the wood one hard enough to drive it against the wall. The wood ball eventually rises to the surface as it rolls along the back wall.

Physical experiments have shown that when two spheres sink in a tank of liquid, one placed just above the other, a phenomenon occurs known as "drafting, kissing, and tumbling." We tested this with the rigid fluid technique and were easily able to reproduce the effect. As shown in figure 5, we placed two spherical stones (radius 8.25cm and relative density 5.7) at the top left of a three meter tall tank of water. Notice that the turquoise stone starts above the marble stone. As they sink together, they drift closer, and eventually tumble around one another. Because of the tumbling, the turquoise stone reaches the bottom first, even though it started on top. We observe qualitatively similar drafting and tumbling for two odd shaped gems in the same virtual tank (figure 6). The detailed motion of these gems is very different from that of the spherical balls, because they have a preferred falling orientation.

In figure 7, eight metal gears of relative density nine are dropped into a one meter wide tank of water. They hit the water at about 4m/s and make a rather large splash. In the water are three wooden bunnies. As the gears sink to the bottom of the tank they strike and interact with the bunnies which are, at the same time, trying to rise to the top. The bunnies eventually reach the surface, though the bunny on the right was lucky to escape the gear that hooked its ears.

To demonstrate the speed of the rigid fluid method we collected computation times for the first second, when most of the action takes place, of the gears and bunnies simulation depicted in figure 7. It took 401 simulation steps to reach the one second mark. The average CPU time per simulation step was 27.5 seconds on a Pentium 4 2GHz with 1GB ram. Just over 95% percent of the CPU time was spent solving equations for the the fluid and level set. Just over 4% of the CPU time was spent in the functions that enable the two-way coupling. Each gear has 250 vertices, and each bunny has 5002 vertices. The computational domain is $64 \times 68 \times 64$.

9 Conclusion and Future Work

We have presented the rigid fluid method as a means of simulating many common two-way fluid-solid coupling scenarios. The main strength of the rigid fluid method lies in the efficient handling of the rigid solids, requiring only a relatively small additional computation on top of that already required by the fluid solver. This remarkably minimal cost follows from the use of distributed Lagrange multipliers [Glowinski et al. 1999] for the solid rigidity constraints, computed with an operator splitting that separately projects onto the divergence-free velocities in the entire domain and the deformation-free velocities inside the solids.

Figure 7: Eight metal gears are thrown down into a pool or water that contains wooden bunnies.

Several fruitful avenues for future work remain. It is possible to do both the divergence-free projection and the rigidity projection in the same step, but the computational cost is substantially higher than doing either projection alone. There are, however, environments that need both projections done at the same time. For example, if the top half of an hour-glass was filled with water and a

rigid ball were to plug the hole separating the top and bottom reservoirs, then both projections, and the rigid body contact constraints, should be done at the same time or water will seep through the plug. If these types of situations are common in a given scene, then we recommend trying an ALE method, or a finite element version of the DLM method [Glowinski et al. 1999].

One simple addition to the method as presented here would be to add joint constraints for our rigid bodies. The α_c and $\mathbf{A}_c$ variables do not need to come from collision and contact forces, they could be used to model the forces necessary for joints, or even human motion controllers. We would like to add these features to our rigid body solver so that simulated divers [Wooten and Hodgins 1996] could splash and interact with the water, and maybe learn to swim.

Objects that occupy very few grid cells are difficult to simulate. A plank of wood is fine as long as it always takes up at least one grid cell along its length, but extremely thin rigid objects, like the wall of a metal bucket, can not be simulated without a sufficiently fine grid, or else the rigidity constraint will not stop water from flowing through the thin walls of the bucket.

10 Acknowledgments

We would especially like to thank Andrew Lackey, M.P.S.E., for creating the beautiful sound effects that accompany our animations. We would also like to thank everyone who spent time reading early versions of this paper, including the reviewers. This research was supported in part financially by NSF grants CCR 36166AR and DMS 0204309, and emotionally by our families. The images in this paper were rendered with PoVRay, available at http://povray.org.

References

CARLSON, M., MUCHA, P. J., VAN HORN, III, R. B., AND TURK, G. 2002. Melting and flowing. In *ACM SIGGRAPH Symposium on Computer Animation*, 167–174.

CARLSON, M. T. 2004. *Rigid, Melting, and Flowing Fluid*. PhD thesis, Georgia Institute of Technology.

CHEN, J. X., AND DA VITORIA LOBO, N. 1995. Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. *Graphical Models and Image Processing 57*, 2, 107–116.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 83–116.

ENRIGHT, D. P., MARSCHNER, S. R., AND FEDKIW, R. P. 2002. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics 22*, 3, 736–744.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 15–22.

FEDKIW, R. P. 2002. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *Journal of Computational Physics 175*, 200–224.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 23–30.

FOSTER, N., AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5, 471–483.

FOSTER, N., AND METAXAS, D. 1997. Controlling fluid animation. In *Proceedings CGI '97*, 178–188.

GÉNEVAUX, O., HABIBI, A., AND DISCHLER, J.-M. 2003. Simulating fluid-solid interaction. In *Graphics Interface*, CIPS, Canadian Human-Computer Communication Society, 31–38.

GLOWINSKI, R., PAN, T.-W., HESLA, T. I., AND JOSEPH, D. D. 1999. A distributed Lagrange multiplier/fictious domain method for particulate flows. *International Journal of Multiphase Flow 25*, 5, 755–794.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. P. 2003. Nonconvex rigid bodies with stacking. *ACM Transactions on Graphics 22*, 3, 871–878.

HIRT, C., AMSDEN, A., AND COOK, J. 1974. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics 14*, 227–253.

O'BRIEN, J. F., ZORDAN, V. B., AND HODGINS, J. K. 2000. Combining active and passive simulations for secondary motion. *IEEE Computer Graphics and Applications 20*, 4, 86–96.

OSHER, S., AND FEDKIW, R. P. 2003. *Level Set Methods and Dynamic Implicit Surfaces*. No. 153 in Applied Mathematical Sciences. Springer-Verlag, New York.

PATANKAR, N. A., SINGH, P., JOSEPH, D. D., GLOWINSKI, R., AND PAN, T.-W. 2000. A new formulation of the distributed Lagrange multiplier/fictious domain method for particulate flows. *International Journal of Multiphase Flow 26*, 9, 1509–1524.

PATANKAR, N. A. 2001. A formulation for fast computations of rigid particulate flows. *Center for Turbulence Research Annual Research Briefs 2001*, 185–196.

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

SINGH, P., HESLA, T. I., AND JOSEPH, D. D. 2003. Distributed Lagrange multiplier method for particulate flows with collisions. *International Journal of Multiphase Flow 29*, 3, 495–509.

STAM, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 121–128.

TAKAHASHI, T., HEIHACHI, U., AND KUNIMATSU, A. 2002. The simulation of fluid-rigid body interaction. In *SIGGRAPH 2002: Sketches & Applications*, 266.

TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., AND UEKI, H. 2003. Realistic animation of fluid with splash and foam. *Computer Graphics Forum 22*, 3, 391–401.

WEIMER, H., AND WARREN, J. 1999. Subdivision schemes for fluid flow. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 111–120.

WITTING, P. 1999. Computational fluid dynamics in a traditional animation environment. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 129–136.

WOOTEN, W. L., AND HODGINS, J. K. 1996. Animation of human diving. *Computer Graphics Forum 15*, 1, 3–14.

WRENNINGE, M. 2003. *Fluid Simulation for Visual Effects*. Master's thesis, Linköpings Universitet, Linköping, Sweden.

YNGVE, G. D., O'BRIEN, J. F., AND HODGINS, J. K. 2000. Animating explosions. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 29–36.