Physics-based Animation of Large-scale Splashing Liquids

Dan Gerszewski
University of Utah

Adam W. Bargteil
University of Utah

Abstract

Fluid simulation has been one of the greatest successes of physics-based animation, generating hundreds of research papers and a great many special effects over the last fifteen years. However, the animation of large-scale, splashing liquids remains challenging. In this paper, we show that a novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries is extremely well-suited to the animation of large-scale, violent, splashing liquids.


Keywords: Unilateral incompressibility, liquid simulation, physics-based animation, fluid-implicit-particle method

1 Introduction

Over the last fifteen years, fluid simulation has emerged as one of the most effective applications of physics-based approaches to animation. The use of fluid simulation for special effects is now commonplace and the topic has received copious attention in the graphics community. Despite tremendous progress, challenges remain. In this paper, we address the problem of large-scale, violent, splashing liquids. We demonstrate that a novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries is extremely well-suited to the animation of such liquids and avoids common artifacts such as artificial surface tension, volume loss/gain, and fluid sticking to obstacles.

Because at the spatial and temporal scales we seek to animate liquids compression is negligible, computer graphics researchers have largely focused on simulating incompressible fluids. Even approaches, such as smoothed particle hydrodynamics (SPH), that are naturally suited to simulate compressible flow are often modified for incompressible flow [Solenthaler and Pajarola 2009]. While computationally efficient, incompressibility induces an artificial surface tension that prevents liquid near the surface from mixing with the surrounding air. This mixing is important at large scales, especially during violent splashes, such as after underwater explosions. In this paper, we show that for single-phase fluid simulation, such mixing is effectively modeled with unilateral incompressibility [Narain et al. 2009; Narain et al. 2010], which allows positive divergence while prohibiting negative pressures, thus avoiding the pressure oscillations found in compressible simulation, while removing the artificial surface tension caused by bilateral incompressibility.

We use a variant of the fluid-implicit-particle (FLIP) method as our underlying simulation method. However, our approach, which we call mass-full FLIP, attaches mass to the particles and more closely resembles compressible FLIP [Brackbill and Ruppel 1986] than the incompressible variety [Zhu and Bridson 2005]. Mass-full FLIP is extremely well-suited to the unilateral incompressibility (UIC) solve. In the context of UIC, ensuring conservation of mass becomes difficult—allowing positive divergence can result in significant volume gain. Like SPH methods, mass-full FLIP conserves mass by conserving particles. Additionally, the UIC is most appropriate in highly turbulent simulations where the numerical viscosity associated with semi-Lagrangian and related schemes would be especially inappropriate.

Finally, we treat obstacles and fluid in a unified manner—we discretize obstacles using particles and rasterize their mass onto the background grid using the same trilinear kernel. We additionally employ the variational approach to obstacles endorsed by Batty and colleagues [2007]. Combined with unilateral incompressibility, our treatment of boundaries easily allows liquids to separate from obstacles, avoiding the common visual artifact of liquid gliding along the ceiling.

We demonstrate our approach on several examples, such as the flooding of a city depicted in Figure 1. Side-by-side comparisons with incompressible simulations clearly demonstrate the different behavior afforded by our approach. In general, the more tumultuous the motion, the more different the results. While we do not expect our approach to replace bilateral incompressibility, we believe the rich behavior afforded by it will prove an important tool for animating large-scale splashing liquids.

2 Related Work

The most closely related work to ours from a technical standpoint is the work of Narain and colleagues, who introduced unilateral incompressibility and applied it to two-dimensional crowd simu-
Recently, Schechter and Bridson [2012] introduced reduced inequality constraints to incorporate boundaries into incom-
jections to achieve high detail. Bodin and colleagues [2012] intro-
to resolve large-scale divergence and then perform finer scale pro-
ulations. Lentine and colleagues [2010] similarly use coarse grids
ing a coarse grid projection to reduce compressibility in SPH sim-
ulations. Raveendran and colleagues [2011] advocated us-
leagues [2009] also used density targeting to reduce fluctuations in
and foam. However, they used these variable densities as targets
ies and are able to achieve fluid expansion in the form of spray
and foam. Their particle system is quite similar to our mass-
leagues [2008], who coupled incompressible flow to a particle sys-
The goal of our method is very similar to that of Losasso and col-
leagues [2008], who coupled incompressible flow to a particle sys-
tem to achieve some of the first really convincing animations of
breaking waves. Their particle system is quite similar to our mass-
full FLIP, though they do not create particles deep in the body of
the fluid. More significantly, like us, they adopted variable densi-
ties and are able to achieve fluid expansion in the form of spray
and foam. However, they used these variable densities as targets
for bilateral incompressibility, whereas we adopt UIC. McAdams
and colleagues [2009] adopted this density targeting approach to
precondition collisions in hair simulation and Solenthaler and col-
leagues [2009] also used density targeting to reduce fluctuations in
SPH simulations. Raveendran and colleagues [2011] advocated us-
ing a coarse grid projection to reduce compressibility in SPH sim-
ulations. Lentine and colleagues [2010] similarly use coarse grids
to resolve large-scale divergence and then perform finer scale pro-
jections to achieve high detail. Bodin and colleagues [2012] intro-
duced inequality constraints to incorporate boundaries into incom-
pressibility solves in SPH fluid simulations.

Recently, Schechter and Bridson [2012] introduced Ghost SPH
to address artificial surface tension in SPH simulations. Their ap-
proach involves using improved boundary conditions through the
introduction of ghost particles. This technique alleviates particle
clumping that results when particles do not have sufficient neigh-
bors to reach their target density—by introducing ghost particles
the target density is reached. Another solution to this problem was
presented by Macklin and Müller [2013]. Our solution is differ-
ent. Instead of modifying the free-surface boundary conditions, We
place a one-sided constraint on the divergence, which is more anal-
ogous to turning off SPH pressure forces if the density were below
the target.

However, the underlying causes of artificial surface tension in SPH
and FLIP simulation are similar, but different. Artificial surface
tension in SPH results from small particle neighborhoods clumping
together to try and achieve a target density, while in grid-based and
FLIP simulations, artificial surface tension is caused by negative
pressures resulting from disallowing positive divergence.

Chentanez and Müller-Fischer [2011b] also set out to create large-
scale effects. To do so in real-time they developed an Eulerian
“tall-cell” simulation system that runs on a GPU. They also incor-
porated important secondary effects including wave textures and
spray, mist, and foam particles. Incorporating such elements into
our approach would likely lead to a much richer visual experience.
The same year they also achieved separation from solid bound-
daries by solving a linear complementarity problem [Chentanez and
Müller 2011a]. They adopted a multi-grid solver and enforced non-
negative pressures only in obstacle boundaries, whereas we use
preconditioned conjugate gradient wrapped in an active-set method
disallow negative pressures in all cells near the liquid surface.

We also draw on the work of Batty and colleagues [2007] for han-
dling boundary conditions. Similar to their work we take into ac-
count the volume occupied by obstacles to adjust the amount of
fluid that can enter a cell. However, while they use a box filter for
obstacles, we use the same trilinear filter for rasterizing obstacles
as we use for particles. We also note that they were the first to cast
wall-separation as a linear complementarity problem, which they
solved with a PATH solver that did not scale to large problems.

There is a rich body of work on fluid simulation in computer graph-
ics. A complete survey is beyond the scope of this paper, but we
whole-heartedly refer the interested reader to the book by Brid-
son [2008] or, for the more mathematically oriented the text by
Chorin and Marsden [2000].

3 Methods

Our approach brings together several components: unilateral in-
compressibility, mass-full FLIP, and blurred obstacles. While some
of these techniques have been employed before inside and outside
of the graphics literature, we demonstrate that their novel combi-
nation is especially effective for computer animation of large-scale
splashing liquids.

3.1 Unilateral Incompressibility

For completeness, we briefly describe unilateral incompressibility
and discuss practical issues in its application to simulating liquids.
The Euler equations describe the motion of inviscid fluids by stating
that mass and momentum are conserved:
\[
\frac{\partial \rho}{\partial t} = - \nabla \cdot (\rho \mathbf{u}) \quad \text{(1)}
\]
\[
\frac{\partial \mathbf{u}}{\partial t} = - \mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \frac{f}{\rho} \quad \text{(2)}
\]

where \( \rho \) denotes the density of the fluid, \( t \) time, \( \mathbf{u} \) the velocity, \( p \) the pressure, and \( f \) external forces such as gravity. For incompressible
flow, the constraint that density be a constant leads to the solution of
a Poisson equation to determine the pressure that will lead to a
divergence-free velocity field, while for compressible flow an equa-
tion of state determines pressure as a function of density [Becker
and Teschner 2007]. In the case of unilaterally incompressible
flows, we place an upper bound on the density of fluid in any given
cell, \( \rho_{\text{max}} \), and require that pressures be non-negative. Addition-
ally, we require that for any given cell either the density is \( \rho_{\text{max}} \)
or that the pressure is zero. Intuitively, this last constraint requires that
the cell be full of liquid or be treated like air.

These constraints can be formulated as a linear complementarity
problem,
\[
A \mathbf{p} + \mathbf{b} \geq 0 \quad \text{(3)}
\]
\[
\mathbf{p} \geq 0 \quad \text{(4)}
\]
\[
\mathbf{p}^T (A \mathbf{p} + \mathbf{b}) = 0 \quad \text{(5)}
\]

with the substitutions
\[
A \mathbf{p} = -\Delta t \nabla \cdot (\mathbf{v} \nabla \mathbf{p}) \quad \text{(6)}
\]
\[
\mathbf{b} = \rho_{\text{max}} - \rho_c + \Delta t \nabla \cdot (\rho_c \mathbf{u}) \quad \text{(7)}
\]

Here, \( \mathbf{v} \) is a face-centered volume fraction (see Section 3.3) repre-
senting the fraction of the volume around the face that may be oc-
cupied by liquid, and \( \rho_c \) and \( \rho_f \) are cell-centered and face-centered
liquid densities, respectively (see Section 3.2). Intuitively, \( \mathbf{b} \) es-
imates the amount of free space (air) in a cell at the end of the timestep
when pressure was zero.

We solve this system using the modified proportioning with re-
duced gradient projections (MPRGP) method as described by
Dostáš [2005; 2009], with the Modified Incomplete Cholesky
(MIC(0)) preconditioner as described by Bridson [2008]. MPRGP
is an active-set method where the active set includes cells where
the pressure is currently zero and the free set contains cells where the pressure is positive. The method interleaves conjugate gradient steps with expansion steps that increase the size of the active set and proportioning steps that add cells to the free set. The method requires an estimate of the induced norm of the matrix (the largest eigenvalue) to use as a bound on step sizes. In practice, this can be found using power iterations. We have found that warm-starting these iterations with the eigenvector from a previous matrix can dramatically reduce the resulting number of matrix multiplications. The method is more general than the problem we have described—it also allows for constraints of the form \( p_i \geq l_i \). We can take advantage of this functionality to allow for negative pressures (suction) away from the surface of the liquid.

After solving for pressures we update the velocity field,

\[
\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - v_f \nabla p_i^n. \tag{8}
\]

It is worth noting that this formulation is not unique. In particular, the volume fractions, \( v_f \) in Equation (8) could be moved to the right hand side and/or the densities, \( p_f \), could be included in the matrix, both of which may seem more natural choices. Just moving the volume fractions to the right hand side leads to instability as the cell-centered volume fractions, needed to multiply \( \rho_{\text{max}} \) and \( \rho_{\text{e}} \), may disagree somewhat with the face-centered volume fractions. Moving the densities into the matrix does not cause instabilities, but leads to fewer expansion steps and less lively motion (see the accompanying video for an example). If the densities are moved to the matrix, moving the face fractions to the right hand side has little effect.

The intuition behind the UIC solve is less straightforward than standard incompressible solves. In a traditional incompressible solve, the solver is prohibited from allowing divergence in any cell that has been labeled liquid. However, with unilateral incompressibility, the solver is free to relabel liquid cells as air by setting the pressure to zero and allowing negative divergence. The presence of zero pressures and negative divergence in the liquid prohibits the sort of long-distance pressure gradients that allow fluid to slosh back and forth. Instead, the fluid quickly settles and comes to rest. Intuitively, our formulation encourages the solver to consider cells labeled liquid as liquid by making them look full, as in an incompressible solve. However, this formulation is more sensitive to spatial oscillations in the density field that naturally occur from numerical errors in particle advection. These oscillations can lead to popping and even explosions, especially when timesteps are large. We make use of levelset values. We first mark each grid face that receives any density. We then sweep over the grid several times. In particular, we combine the trilinear interpolated velocity from the grid (PIC) and FLIP to update the particle velocities. That is, we apply external forces (gravity) and velocities and masses onto the grid. Velocity and mass contributions from each particle are accumulated onto each face; just mass is contributed to cell centers. We use the standard trilinear weighting kernel. After all particles have contributed, velocities are normalized and masses are converted to densities by dividing by the volume of a cell. Specifically,

\[
m_f = m_p \sum_p T(x_p - x_f), \tag{10}
\]

\[
u_f = \frac{m_p}{m_f} \sum_p u_p T(x_p - x_f), \tag{11}
\]

\[
\rho_f = \frac{m_f}{h^3}, \tag{12}
\]

where \( m_f \) is the mass of a face, \( m_p \) is the mass of a particle, \( T(\cdot) \) is the trilinear interpolation kernel, \( x_p \) is the particle position, \( x_f \) is the position of the center of a face, \( u_p \) and \( u_f \) are the \( u \)-components of the velocity of the particle and face, respectively (similar equations exist for the \( v \)- and \( w \)-components), \( \rho_f \) is the density at the face center, and \( h \) is the grid spacing. Note we do not need to explicitly store both \( m_f \) and \( \rho_f \). Because we divide by density in Equation (8), any cell that has mass less than a threshold is treated as air and not included in the unilateral incompressibility solve.

**Velocity Extrapolation** We perform a simple velocity extrapolation algorithm that is similar in spirit to fast sweeping, but does not make use of levelset values. We first mark each grid face that received any density. We then sweep over the grid several times. In each sweep a face that has marked neighboring faces is assigned an average of the neighboring velocities and is itself marked. This approach is less accurate than building a levelset and applying fast sweeping, but is much faster.

**Update Particle Velocity** As is commonly done, we use a combination of PIC and FLIP to update the particle velocities. That is we combine the trilinear interpolated velocity from the grid (PIC) with the change in grid velocity (FLIP).

### 3.2 Mass-full FLIP

Unilateral incompressibility and mass-full FLIP are very well-suited to each other. Unilateral incompressibility, like bilateral incompressibility, requires the solution of a global system every timestep. This solve is easily performed on mass-full FLIP’s background grid. Moreover, we require density estimates at various locations on the grid. These are easily obtained by rasterizing particle mass onto the grid and dividing by volume. Furthermore, mass-full FLIP, like SPH, automatically conserves mass by conserving the number of particles—a very attractive feature. Finally, FLIP’s advection scheme results in very low dissipation, perfect for our target of large-scale splashing liquids.

Given the popularity of FLIP in computer graphics we only briefly describe the method (for details please see [Zhu and Bridson 2005; Bridson 2008]). The anatomy of a timestep in our system is as follows:

1. Rasterize particle velocities and masses onto the grid
2. Apply external forces (gravity)
3. Solve unilateral incompressibility and apply pressure gradient (see Section 3.1)
4. Extrapolate velocity field
5. Update particle velocities
6. Advect particles through velocity field

**Particle Rasterization** In the first step, we rasterize particle velocities and masses onto the grid. Velocity and mass contributions from each particle are accumulated onto each face; just mass is contributed to cell centers. We use the standard trilinear weighting kernel. After all particles have contributed, velocities are normalized and masses are converted to densities by dividing by the volume of a cell. Specifically,

\[
m_f = m_p \sum_p T(x_p - x_f), \tag{10}
\]

\[
u_f = \frac{m_p}{m_f} \sum_p u_p T(x_p - x_f), \tag{11}
\]

\[
\rho_f = \frac{m_f}{h^3}, \tag{12}
\]

where \( m_f \) is the mass of a face, \( m_p \) is the mass of a particle, \( T(\cdot) \) is the trilinear interpolation kernel, \( x_p \) is the particle position, \( x_f \) is the position of the center of a face, \( u_p \) and \( u_f \) are the \( u \)-components of the velocity of the particle and face, respectively (similar equations exist for the \( v \)- and \( w \)-components), \( \rho_f \) is the density at the face center, and \( h \) is the grid spacing. Note we do not need to explicitly store both \( m_f \) and \( \rho_f \). Because we divide by density in Equation (8), any cell that has mass less than a threshold is treated as air and not included in the unilateral incompressibility solve.

**Velocity Extrapolation** We perform a simple velocity extrapolation algorithm that is similar in spirit to fast sweeping, but does not make use of levelset values. We first mark each grid face that received any density. We then sweep over the grid several times. In each sweep a face that has marked neighboring faces is assigned an average of the neighboring velocities and is itself marked. This approach is less accurate than building a levelset and applying fast sweeping, but is much faster.

**Update Particle Velocity** As is commonly done, we use a combination of PIC and FLIP to update the particle velocities. That is we combine the trilinear interpolated velocity from the grid (PIC) with the change in grid velocity (FLIP).
Particle Advection We use the second-order trapezoidal rule or first order Euler integration for advection. In the later case the extrapolation step may be skipped. Particle paths are clipped against domain boundaries and the levelset representation of obstacles. In the later case, we try stepping along the gradient of the levelset to place the particle on the obstacle surface. If this fails to converge (or places the particle outside the domain boundary), we perform several bisection steps along the particle’s initial path to find a position with a positive (outside the obstacle), but small, levelset value.

3.3 Obstacle Handing

Our handling of obstacles is the most novel component of our approach. We sample the obstacles with particles, just as we do the liquid. We then rasterize obstacle particles onto the grid using the trilinear interpolation weights we also use for the particles. In effect we are blurring the boundaries of the obstacle onto the grid using the trilinear interpolation weights. We then rasterize obstacle particles onto the grid using the trilinear interpolation weights we also use for the particles. 

As our unilateral incompressibility solve does not allow negative pressures, there is no “suction” along obstacles and we automatically obtain wall-separating boundary conditions as cells next to obstacles “expand.” That is, the pressure solve allows the fluid velocity field to point out of the obstacle, but not into it once the maximum density is reached. Additionally, as noted above, during advection we ensure that particles are outside obstacle levelsets.

It is the novel combination of blurred obstacle boundaries, unilateral incompressibility, variational volume fractions and particle collision detection that allows our method to handle obstacles seamlessly—with no special effort we get wall-separation. These benefits do come at a cost, however. We sacrifice sharp boundary conditions and sharp interfaces. This sacrifice is justified in the context of large-scale single-phase liquid simulations, but would prove too great for multiphase flow or in cases where the boundary layer plays a key role.

4 Results and Discussion

We include two comparisons between our approach and an incompressible FLIP solver. Like the solver detailed above, our incompressible FLIP uses the second-order trapezoidal rule, which produces somewhat smoother results than the commonly used midpoint method, and our simple velocity extrapolation. One difference is that while our particles will contribute mass to nearby cells due to the trilinear filter, our incompressible solver uses a “box filter” to determine which cells are labeled “liquid.” In our experience, this diminished volume gain and led to better results from particle skinning. Our incompressible solver also uses traditional non-blurred obstacles.

In the first comparison we set up two fountains (see Figure 2). Our approach allows the liquid to separate and expand in a natural manner, while the artificial surface tension of the incompressible solver causes the fountain to oscillate and collapse on itself. The second comparison is a dam break with an obstacle (see Figure 3). In this example our approach creates a large splash as the liquid passes over the obstacle, while the incompressible approach flows over the obstacle with almost no noticeable splash. This example also demonstrates incompressible FLIP’s tendency toward volume gain as a single particle can force a cell to be labeled liquid. Our additional examples (see Figures 4, 5, 6, and 7) demonstrate the capabilities and large-scale, splashy fluid effects. Our simulations were performed with a variety of grid resolutions and domain scales, see Table 1 for details and timing results.

Limitations and Future Work A primary limitation of our approach is that there is no representation of the liquid surface, forcing us to rely on particle skinning approaches, which can lead to artifacts, especially at low resolutions and with uniform or overly randomized samplings. Interestingly, while other researchers using FLIP have generally favored lower numbers of particles per cell, we found that increasing the number of particles per cell was a very effective strategy for achieving higher resolution animations without requiring the solution of larger LCPs. An alternative to particle skinning would be to couple our particles to an explicit mesh surface as done by Yu and colleagues [2012], though this would not allow for the frame-level parallelism of our particle skinning approach. Another disadvantage of our approach is the necessity for two LCP solves, both of which are more expensive than the simple
Table 1: Grid scale, resolutions, and timing information for all examples in this paper. Timing results are given in average seconds for one 30 Hz frame.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Timesteps per frame</th>
<th>Grid spacing (h)</th>
<th>Domain size</th>
<th>Particle count</th>
<th>Total time</th>
<th>Solve time</th>
<th>Extrapolate velocity</th>
<th>Particle advection</th>
<th>Particle rasterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>3</td>
<td>2</td>
<td>250x80x80</td>
<td>16307596</td>
<td>75.6</td>
<td>35.3</td>
<td>1.5</td>
<td>23.5</td>
<td>14.4</td>
</tr>
<tr>
<td>Figure 2 (UIC)</td>
<td>12</td>
<td>0.05</td>
<td>50x100x50</td>
<td>104988</td>
<td>7.8</td>
<td>5.1</td>
<td>0.8</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Figure 3 (UIC)</td>
<td>11</td>
<td>0.1</td>
<td>80x60x40</td>
<td>1539000</td>
<td>21.7</td>
<td>5.2</td>
<td>0.5</td>
<td>9</td>
<td>6.8</td>
</tr>
<tr>
<td>Figure 4</td>
<td>12</td>
<td>0.1</td>
<td>100x100x100</td>
<td>5508794</td>
<td>118.5</td>
<td>54.2</td>
<td>2.7</td>
<td>30.3</td>
<td>29.9</td>
</tr>
<tr>
<td>Figure 5</td>
<td>12</td>
<td>0.1</td>
<td>100x120x100</td>
<td>2060247</td>
<td>81.6</td>
<td>52.1</td>
<td>4.6</td>
<td>12.4</td>
<td>10.2</td>
</tr>
<tr>
<td>Figure 7</td>
<td>11</td>
<td>0.125</td>
<td>208x112x160</td>
<td>6937747</td>
<td>556</td>
<td>470</td>
<td>12.6</td>
<td>39.1</td>
<td>28.7</td>
</tr>
<tr>
<td>Figure 6</td>
<td>3</td>
<td>0.0625</td>
<td>80x80x80</td>
<td>3430944</td>
<td>20</td>
<td>11.3</td>
<td>0.3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 4: A quadruple dam break creates a huge splash in the center of the scene.

Figure 5: Several liquid object fall into a circular pool of water.

Figure 6: A series of underwater explosion cause large-scale splashing.

Figure 7: A large dam break above uneven terrain.

Poisson solve in traditional incompressibility. We would also like to improve the computational efficiency of our approach. Many software design decisions were made favoring ease of debugging and experimentation over efficiency, so we believe there is much room for improvement. We also would like to experiment with alternative LCP solvers, such as the multigrid method of Chentanez and Müller [2011a] and interior point methods.

One, perhaps subtle, disadvantage of assigning mass to the particles is that it makes reseeding very difficult. Placing too many particles in a cell will result in unwanted expansion, while too few will result in collapse. It took several iterations to produce the example in Figure 2, which required solving for the number of particles to add based on the fountain velocity and simulation timestep. One potential approach to address this drawback would be to use variable particle masses, though this could easily lead to other complications and artifacts. Finally, we note that our approach essentially blurs both the liquid surface and obstacles. While this is acceptable for large-scale, splashy behavior, in many contexts sharp boundaries are essential to compute the desired behavior [Pfaff et al. 2009].

Our novel combination of unilateral incompressibility, mass-full FLIP, and blurred boundaries provides a very effective simulation strategy for large-scale splashing liquids. By avoiding the artificial surface tension of traditional incompressibility our approach is able to simulate liquids that mix freely with the surrounding air, while also avoiding the oscillations present in smoothed particle hydrodynamics. The particle-based mass-full FLIP is well-suited to splashes and the thin-sheets they create and does not suffer from mass loss or gain. Our blurred boundaries unify the liquid and obstacle representations and work with the unilateral incompressibility to allow liquid to detach from obstacles. Overall, we believe our approach offers a number of advantages over the state-of-the-art for animating large-scale, splashy liquids.
Acknowledgments

The author thank Nafees Bin Zafar, Kenny Erleben, Peter-Pike Sloan, and the anonymous reviewers for helpful feedback. We also thank Mike Farnsworth and Solid Angle for donating licenses to their Arnold renderer and the Scientific Computing and Imaging Institute for sharing their cluster. This work was supported in part by National Science Foundation awards IIS-1249756 and CNS-0855167 and a gift from Adobe Systems Incorporated.

References


