Animating Sand as a Fluid

Yongning Zhu[*]
University of British Columbia

Robert Bridson[*]
University of British Columbia

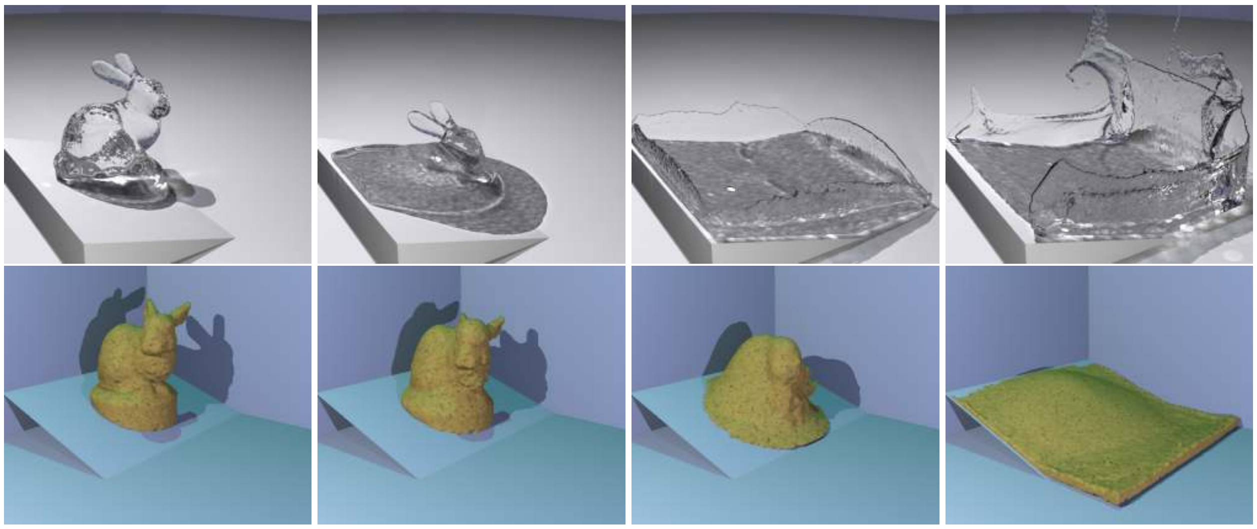

Figure 1: The Stanford bunny is simulated as water and as sand.

Abstract

We present a physics-based simulation method for animating sand. To allow for efficiently scaling up to large volumes of sand, we abstract away the individual grains and think of the sand as a continuum. In particular we show that an existing water simulator can be turned into a sand simulator with only a few small additions to account for inter-grain and boundary friction.

We also propose an alternative method for simulating fluids. Our core representation is a cloud of particles, which allows for accurate and flexible surface tracking and advection, but we use an auxiliary grid to efficiently enforce boundary conditions and incompressibility. We further address the issue of reconstructing a surface from particle data to render each frame.

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

Keywords: sand, water, animation, physical simulation

1 Introduction

The motion of sand—how it flows and also how it stabilizes—has transfixed countless children at the beach or playground. More generally, granular materials such as sand, gravel, grains, soil, rubble, flour, sugar, and many more play an important role in the world, thus we are interested in animating them.

[*]email: {yzhu, rbridson}@cs.ubc.ca

For small numbers of grains (say up to a few thousand) it is possible to simulate each one as an individual rigid body, but scaling this up to just a handful of sand is clearly infeasible; a sand dune containing trillions of grains is out of the question. Thus we take a continuum approach, in essence squinting our eyes and pretending there are no separate grains but rather the sand is a continuous material. The question is then what the constitutive laws of this continuum should be: how should it respond to force?

The dynamics of granular materials have been studied extensively in the field of soil mechanics, but for the purposes of plausible animation we are willing to simplify their models drastically to allow efficient implementation. In section 3 we detail this simplification, arriving at a model which may be implemented by adding just a little code to an existing water simulation.

We also present a new fluid simulation method in section 4. As previous papers on simulating fluids have noted, grids and particles have complementary strengths and weaknesses. Here we combine the two approaches, using particles for our basic representation and for advection, but auxiliary grids to compute all the spatial interactions (e.g. boundary conditions, incompressibility, and in the case of sand, friction forces).

Our simulations only output particles that indicate where the fluid is. To actually render the result, we need to reconstruct a smooth surface that wraps around these particles. Section 5 explains our work in this direction, and some of the advantages this framework has over existing approaches.

2 Related Work

We briefly review relevant papers, primarily in graphics, that provide a context for our research. In later sections we will refer to those that explicitly guided our work.

Several authors have worked on sand animation, beginning with Miller and Pearce[1989] who demonstrated a simple particle system model with short-range interaction forces which could be tuned to achieve (amongst other things) powder-like behavior. Later Luciani et al.[1995] developed a similar particle system model specifically for granular materials using damped nonlinear springs at a coarse length scale and a faster linear model at a fine length scale. Li and Moshell[1993] presented a dynamic height-field simulation of soil based on the standard engineering Mohr-Coulomb constitutive model of granular materials. Sumner et al.[1998] took a similar heightfield approach with simple displacement and erosion rules to model footprints, tracks, etc. Onoue and Nishita[2003] recently extended this to multi-valued heightfields, allowing for some 3D effects.

As we mentioned above, one approach to granular materials is directly simulating the grains as interacting rigid bodies. Milenkovic[1996] demonstrated a simulation of 1000 rigid spheres falling through an hour-glass using optimization methods for resolving contact. Milenkovic and Schmidl[2001] improved the capability of this algorithm, and Guendelman et al.[2003] further accelerated rigid body simulation, showing 500 nonconvex rigid bodies falling through a funnel.

Particles have been a popular technique in graphics for water simulation. Desbrun and Cani[1996] introduced Smooth Particle Hydrodynamics (see Monaghan[1992] for the standard review of SPH) to the animation literature, demonstrating a viscous fluid flow simulation using particles alone. Müller et al.[2003] developed SPH further for water simulation, achieving impressive interactive results. Premoze et al.[2003] used a variation on SPH with an approximate projection solve for their water simulations, but generated the final rendering geometry with a grid-based level set. Particles were also used by Takeshita et al.[2003] for fire.

There has been more work on animating water with grids, however. Foster and Metaxas[1996] developed the first grid-based fully 3D water simulation in graphics. Stam[1999] provided the semi-Lagrangian advection method for faster simulation, but whose excessive numerical dissipation was later mitigated by Fedkiw et al.[2001] with higher order interpolation and vorticity confinement. Foster and Fedkiw[2001] incorporated this into a water solver, and added a level set—augmented by marker particles to counter mass loss—for high quality surface tracking. To this model Génevaux et al.[2003] added two-way fluid-solid interaction forces. Carlson et al.[2002] added variable viscosity to a water solver, providing beautiful simulations of wet sand dripping (achieved simply by increased viscosity). Enright et al.[2002b] extended the Foster and Fedkiw approach with particles on both sides of the interface and velocity extrapolation into the air. Losasso et al.[2004] extended this to dynamically adapted octree grids, providing much enhanced resolution, and Rasmussen et al.[2004] improved the boundary conditions and velocity extrapolation while adding several other features important for visual effects production. Carlson et al.[2004] added better coupling between fluid and rigid body simulations—which we parenthetically note had its origins in scientific work on simulating flow with sand grains. Hong and Kim[2003] and Takahashi et al.[2003] introduced volume-of-fluid algorithms for animating multi-phase flow (water *and* air, as opposed to just the water of the free-surface flows animated above).

For more general plastic flow, most of the graphics work has dealt with meshes. Terzopoulos and Fleischer[1988] first introduced plasticity to physics-based animation, with a simple 1D model applied to their springs. O'Brien et al.[2002] added plasticity to a fracture-capable tetrahedral-mesh finite element simulation to support ductile fracture. Irving et al.[2004] introduced a more sophisticated plasticity model along with their invertible finite elements, also for tetrahedral meshes. Real-time elastic simulation including plasticity has been the focus of several papers (e.g. [Müller and Gross 2004]). Closest in spirit to the current paper, Goktekin et al.[2004] added elastic forces and associated plastic flow to a water solver, enabling animation of a wide variety of non-Newtonian fluids.

For a scientific description of the physics of granular materials, we refer the reader to Jaeger et al.[1996]. In the soil mechanics literature there is a long history of numerically simulating granular materials using a elasto-plastic finite element formulation, with Mohr-Coulomb or Drucker-Prager yield conditions and various non-associated plastic flow rules. We highlight the standard work of Nayak and Zienkiewicz[1972], and the book by Smith and Griffiths[1998] which contains code and detailed comments on FEM simulation of granular materials. Landslides and avalanches are two granular phenomena of particular interest, and generally have been studied using depth-averaged 2D equations introduced by Savage and Hutter[1989]. For alternatives that simulate the material at the level of individual grains, Herrmann and Luding's review[1998] is a good place to start.

Our new fluid code traces its history back to the early particle-in-cell (PIC) work of Harlow[1963] for compressible flow. Shortly thereafter Harlow and Welch[1965] invented the marker-and-cell method for incompressible flow, with its characteristic staggered grid discretization and marker particles for tracking a free surface—the other fundamental elements of our algorithm. PIC suffered from excessive numerical dissipation, which was cured by the Fluid-Implicit-Particle (FLIP) method[Brackbill and Ruppel 1986]; Kothe and Brackbill[1992] later worked on adapting FLIP to incompressible flow. Compressible FLIP was also extended to a elasto-plastic finite element formulation, the Material Point Method[Sulsky et al. 1995], which has been used to model granular materials at the level of individual grains[Bardenhagen et al. 2000] amongst other things. MPM was used by Konagai and Johansson[2001] for simulating large-scale (continuum) granular materials, though only in 2D and at considerable computational expense.

3 Sand Modeling

3.1 Frictional Plasticity

The standard approach to defining the continuum behavior of sand and other granular (or "frictional") materials is in terms of plastic yielding. Suppose the stress tensor σ has mean stress $\sigma_m = tr(\sigma)/3$ and Von Mises equivalent shear stress $\bar{\sigma} = |\sigma - \sigma_m \delta|_F/\sqrt{2}$ (where $|\cdot|_F$ is the Frobenius norm). The Mohr-Coulomb[1] law says the material will not yield as long as

$$\sqrt{3}\bar{\sigma} < \sin\phi\,\sigma_m \qquad (1)$$

where ϕ is the friction angle. Heuristically, this is just the classic Coulomb static friction law generalized to tensors: the shear stress $\sigma - \sigma_m\delta$ that tries to force particles to slide against one another is annulled by friction if the pressure σ_m forcing the particles against each other is proportionally strong enough. The coefficient of friction μ commonly used in Coulomb friction is replaced here by $\sin\phi$.

[1]Technically Mohr-Coulomb uses the Tresca definition of equivalent shear stress, but since it is not smooth and poses numerical difficulties, most people use Von Mises.

If the yield condition is met and the sand can start to flow, a flow rule is required. The simplest reasonable flow direction is $\sigma - \sigma_m \delta$. Heuristically this means the sand is allowed to slide against itself in the direction that the shearing force is pushing. Note that this is nonassociated, i.e. not equal to the gradient of the yield condition. While associated flow rules are simpler and are usually assumed for plastic materials (and have been used exclusively before in graphics[O'Brien et al. 2002; Goktekin et al. 2004; Irving et al. 2004]), they are impossible here. The Mohr-Coulomb law compares the shear part of the stress to the dilatational part, unlike normal plastic materials which ignore the dilation part. Thus an associated flow rule would allow the material to shear if and only if it dilates proportionally—the more the sand flows, the more it expands. Technically sand does dilate a little when it begins to flow—the particles need some elbow room to move past each other—but this dilation stops as soon as the sand is freely flowing. This is a fairly subtle effect that is very difficult to observe with the eye, so we confidently ignore it for animation.

Once plasticity has been defined, the standard soil mechanics approach to simulation would be to add a simple elasticity relation, and use a finite element solver to simulate the sand. However, we would prefer to avoid the additional complexity and expense of FEM (which would require numerical quadrature, an implicit solver to avoid stiff time step restrictions due to elasticity, return mapping for plasticity, etc.). For the purposes of plausible animation we will make some further simplifying assumptions so we can retrofit sand modeling into an existing water solver.

3.2 A Simplified Model

We first ignore the nearly imperceptible elastic deformation regime (due to rock grains deforming) and the tiny volume changes that sand undergoes as it starts and stops flowing. Thus our domain can be decomposed into regions which are rigidly moving (where the Mohr-Coulomb yield condition has not been met) and the rest where we have an incompressible shearing flow.

We then further assume that the pressure required to make the entire velocity field incompressible will be similar to the true pressure in the sand. This is certainly false: for example, it is well known that a column of sand in a silo has a maximum pressure independent of the height of the column[2] whereas in a column of water the pressure increases linearly with height. However, we can only think of one case where this might be implausible in animation: the pressure limit means sand in an hour glass flows at a constant rate independent of the amount of sand remaining (which is the reason hour glasses work). We suggest our inaccurate results will still be plausible in most other cases.

Since we are neglecting elastic effects, we assume that where the sand is flowing the frictional stress is simply

$$\sigma_f = -\sin\phi\, p \frac{D}{\sqrt{1/3}|D|_F} \tag{2}$$

where $D = (\nabla u + \nabla u^T)/2$ is the strain rate. That is, the frictional stress is the point on the yield surface that most directly resists the sliding.

We finally need a way of determining the yield condition (when the sand should be rigid) without tracking elastic stress and strain.

[2]This is due to the force that resists the weight of the sand above being transferred to the walls of the silo by friction—sometimes causing silos to unexpectedly collapse as a result.

Consider one grid cell, of side length Δx. The relative sliding velocities between opposite faces in the cell are $V_r = \Delta x D$. The mass of the cell is $M = \rho\Delta x^3$. Ignoring other cells, the forces required to stop all sliding motion in a time step Δt are $F = -V_r M/\Delta t = -(\Delta x D)(\rho\Delta x^3)/\Delta t$, derived from integrating Newton's law to get the update formula $V^{new} = V + \Delta t F/M$. Dividing F by the cross-sectional area Δx^2 to get the stress gives:

$$\sigma_{rigid} = -\frac{\rho D \Delta x^2}{\Delta t} \tag{3}$$

We can check if this satisfies our yield inequality: if it does (i.e. if the sand can resist forces and inertia trying to make it slide), then the material in that cell should be rigid at the end of the time step.

This gives us the following algorithm. Every time step, after advection, we do the usual water solver steps of adding gravity, applying boundary conditions, solving for pressure to enforce incompressibility, and subtracting $\Delta t/\rho\nabla p$ from the intermediate velocity field to make it incompressible. Then we evaluate the strain rate tensor D in each grid cell with standard central differences. If the resulting σ_{rigid} from equation 3 satisfies the yield condition (using the pressure we computed for incompressibility) we mark the grid cell as rigid and store σ_{rigid} at that cell. Otherwise, we store the sliding frictional stress σ_f from equation 2. We then find all connected groups of rigid cells, and project the velocity field in each separate group to the space of rigid body motions (i.e. find a translational and rotational velocity which preserve the total linear and angular momentum of the group). The remaining velocity values we update with the frictional stress, using standard central differences for $u+ = \Delta t/\rho\nabla\cdot\sigma_f$.

3.3 Frictional Boundary Conditions

So far we have covered the friction within the sand, but there is also the friction between the sand and other objects (e.g. the solid wall boundaries) to worry about. Our simple solution is to use the friction formula of Bridson et al.[2002] when we enforce the $u \cdot n \geq 0$ boundary condition on the grid. When we project out the normal component of velocity, we reduce the tangential component of velocity proportionately, clamping it to zero for the case of static friction:

$$u_T = \max\left(0, 1 - \frac{\mu|u\cdot n|}{|u_T|}\right) u_T \tag{4}$$

where μ is the Coulomb friction coefficient between the sand and the wall.

This is a crucial addition to a fluid solver, since the boundary conditions previously discussed in the animation literature either don't permit any kind of sliding ($u = 0$) which means sand sticks even to vertical surfaces, or always allow sliding ($u \cdot n \geq 0$, possibly with a viscous reduction of tangential velocity) which means sand piles will never be stable. Compare figure 2, which includes friction, to figure 1 which doesn't.

3.4 Cohesion

A common enhancement to the basic Mohr-Coulomb condition is the addition of cohesion:

$$\sqrt{3}\bar{\sigma} < \sin\phi\,\sigma_m + c \tag{5}$$

where $c > 0$ is the cohesion coefficient. This is appropriate for soils or other slightly sticky materials that can support some tension before yielding.

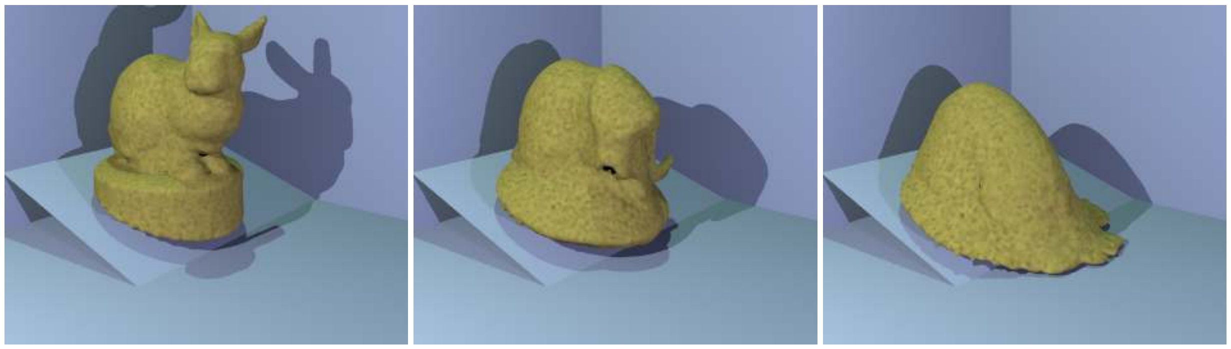

Figure 2: The sand bunny with a boundary friction coefficient $\mu = 0.6$: compare to figure 1 where zero boundary friction was used.

We have found that including a very small amount of cohesion improves our results for supposedly cohesion-less materials like sand. In what should be a stable sand pile, small numerical errors can allow some slippage; a tiny amount of cohesion counters this error and makes the sand stable again, without visibly effecting the flow when the sand should be moving.

However, for modeling soils with their true cohesion coefficient, our method is too stable: obviously unstable structures are implausibly held rigid. We will investigate this issue in the future.

4 Fluid Simulation Revisited

4.1 Grids and Particles

There are currently two main approaches to simulating fully three-dimensional water in computer graphics: Eulerian grids and Lagrangian particles. Grid-based methods (recent papers include [Carlson et al. 2004; Goktekin et al. 2004; Losasso et al. 2004]) store velocity, pressure, some sort of indicator of where the fluid is or isn't, and any additional variables on a fixed grid. Usually a staggered "MAC" grid is used, allowing simple and stable pressure solves. Particle-based methods, exemplified by Smoothed Particle Hydrodynamics (see [Monaghan 1992; Müller et al. 2003; Premoze et al. 2003] for example), dispense with grids except perhaps for accelerating geometric searches. Instead fluid variables are stored on particles which represent actual chunks of fluid, and the motion of the fluid is achieved simply by moving the particles themselves. The Lagrangian form of the Navier-Stokes equations (sometimes using a compressible formulation with a fictitious equation of state) is used to calculate the interaction forces on the particles.

The primary strength of the grid-based methods is the simplicity of the discretization and solution of the incompressibility condition, which forms the core of the fluid solver. Unfortunately, grid-based methods have difficulties with the advection part of the equations. The currently favored approach in graphics, semi-Lagrangian advection, suffers from excessive numerical dissipation due to accumulated interpolation errors. To counter this nonphysical effect, a nonphysical term such as vorticity confinement must be added (at the expense of conserving angular or even linear momentum). While high resolution methods from scientific computing can accurately solve for the advection of a well-resolved velocity field through the fluid, their implementation isn't entirely straightforward: their wide stencils make life difficult on coarse animation grids that routinely represent features only one or two grid cells thick. Moreover, even fifth-order accurate methods fail to accurately advect level sets[Enright et al. 2002a] as are commonly used to represent the water surface, quickly rounding off any small features. The most attractive alternative to level sets is volume-of-fluid (VOF), which conserves water up to floating-point precision: however it has difficulties maintaining an accurate but sharply-defined surface. Coupling level sets with VOF is possible[Sussman 2003] but at a significant implementation and computational cost.

On the other hand, particles trivially handle advection with excellent accuracy—simply by letting them flow through the velocity field using ODE solvers—but have difficulties with pressure and the incompressibility condition, often necessitating smaller than desired time steps for stability. SPH methods in particular can't tolerate nonuniform particle spacing, which can be difficult to enforce throughout the entire length of a simulation.

Realizing the strengths and weaknesses of the two approaches complement each other Foster and Fedkiw[2001] and later Enright et al.[2002b] augmented a grid-based method with marker particles to correct the errors in the grid-based level set advection. This particle-level set method, most recently implemented on an adaptive octree[Losasso et al. 2004], has produced the highest fidelity water animations in the field. However, we believe particles can be exploited even further, simplifying and accelerating the implementation, and affording some new benefits in flexibility.

4.2 Particle-in-Cell Methods

Continuing the graphics tradition of recalling early computational fluid dynamics research, we return to the Particle-in-Cell (PIC) method of Harlow[1963]. This was an early approach to simulating compressible flow that handled advection with particles, but everything else on a grid. At each time step, the fluid variables at a grid point were initialized as a weighted average of nearby particle values, then updated on the grid with the non-advection part of the equations. The new particle values were then interpolated from the updated grid values, and finally the particles moved according to the grid velocity field.

The major problem with PIC was the excessive numerical diffusion caused by repeatedly averaging and interpolating the fluid variables. Brackbill and Ruppel[1986] cured this with the Fluid-Implicit-Particle (FLIP) method, which achieved "an almost total absence of numerical dissipation and the ability to represent large variations in data." The crucial change was to make the particles the fundamental representation of the fluid, and use the auxiliary grid simply to **increment** the particle variables according to the change computed on the grid.

We have adapted PIC and FLIP to incompressible flow[3] as follows:

- Initialize particle positions and velocities
- For each time step:
  - At each staggered MAC grid node, compute a weighted average of the nearby particle velocities
  - For FLIP: Save the grid velocities.
  - Do all the non-advection steps of a standard water simulator on the grid.
  - For FLIP: Subtract the new grid velocities from the saved velocities, then add the interpolated difference to each particle.
  - For PIC: Interpolate the new grid velocity to the particles.
  - Move particles through the grid velocity field with an ODE solver, making sure to push them outside of solid wall boundaries.
  - Write the particle positions to output.

Observe there is no longer any need to implement grid-based advection, or the matching schemes such as vorticity confinement and particle-level set to counter numerical dissipation.

4.2.1 Initializing Particles

We have found a reasonable effective practice is to create 8 particles in every grid cell, randomly jittered from their $2 \times 2 \times 2$ subgrid positions to avoid aliasing when the flow is underresolved at the simulation resolution. With less particles per grid cell, we tend to run into occasional "gaps" in the flow, and more particles allow for too much noise. To help with surface reconstruction later (section 5) we reposition particles that lie near the initial water surface (say within one grid cell width) to be exactly half a grid cell away from the surface.

4.2.2 Transferring to the Grid

Each grid point takes a weighted average of the nearby particles. We define "near" as being contained in the cube of twice the grid cell width centered on the grid point. (Note that on a staggered MAC grid, the different components of velocity will have offset neighborhoods.) The weight of a particle in this cube is the standard trilinear weighting. We also mark the grid cells that contain at least one particle. In future work we will investigate using a more accurate indicator, e.g. reconstructing a signed distance field on the grid from distance values stored on the particles. This would allow us to easily implement a second-order accurate free surface boundary condition[Enright et al. 2003] which significantly reduces grid artifacts.

We also note that the grid in our simulation is purely an auxiliary structure to the fundamental particle representation. In particular, we are not required to use the same grid every time step. An obvious optimization which we have not yet implemented—rendering is currently our bottleneck—is to define the grid according to the bounding box of the particles every time step. This also means we have no a priori limits on the simulation domain. We plan in the future to detect when clusters of particles have separated and use separate bounding grids for them, for significantly improved efficiency.

[3]The one precedent for this that we could find is a reference to an unpublished manuscript[Kothe and Brackbill 1992].

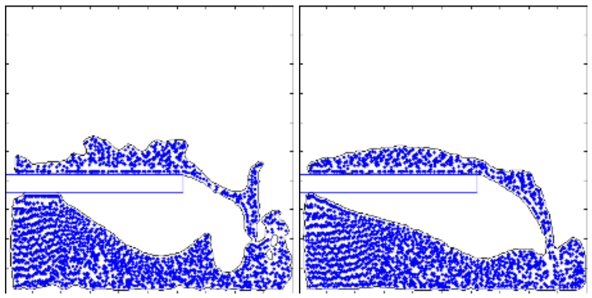
Figure 3: FLIP vs. PIC velocity update for the same simulation. Notice the small-scale velocities preserved by FLIP but smoothed away by PIC.

4.2.3 Solving on the Grid

We first add the acceleration due to gravity to the grid velocities. We then construct a distance field $\phi(x)$ in the unmarked (non-fluid) part of the grid using fast sweeping[Zhao 2005] and use that to extend the velocity field outside the fluid with the PDE $\nabla u \cdot \nabla \phi = 0$ (also solved with fast sweeping). We enforce boundary conditions and incompressibility as in Enright et al.[2002b], then extend the new velocity field again using fast sweeping (as we found it to be marginally faster and simpler to implement than fast marching[Adalsteinsson and Sethian 1999]).

4.2.4 Updating Particle Velocities

At each particle, we trilinearly interpolate either the velocity (PIC) or the change in velocity (FLIP) recorded at the surrounding eight grid points. For viscous flows, such as sand, the additional numerical viscosity of PIC is beneficial; for inviscid flows, such as water, FLIP is preferable. See figure 3 for a 2D view of the differences between PIC and FLIP. A weighted average of the two can be used to tune just how much numerical viscosity is desired (of course, a viscosity step on the grid in concert with PIC would be required for highly viscous fluids).

4.2.5 Moving Particles

Once we have a velocity field defined on the grid (extrapolated to exist everywhere) we can move the particles around in it. We use a simple RK2 ODE solver with five substeps each of which is limited by the CFL condition (so that a particle moves less than one grid cell in each substep), similar to Enright et al.[2002b]. We additionally detect when particles penetrate solid wall boundaries due simply to truncation error in RK2 and the grid-based velocity field, and move them in the normal direction back to just outside the solid, to avoid the occasional stuck-particle artifact this would otherwise cause.

5 Surface Reconstruction from Particles

Our simulations output the positions of the particles that define the location of the fluid. For high quality rendering we need to reconstruct a surface that wraps around the particles. Of course we can give up on direct reconstruction, e.g. running a level set simulation guided by the particles[Premoze et al. 2003]. While this is an effective solution, we believe a fully particle-based reconstruction can have significant advantages.

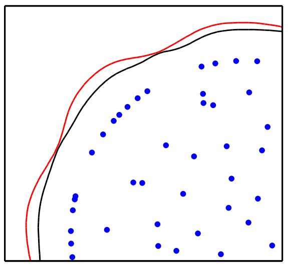

Figure 4: Comparison of our implicit function and blobbies for matching a perfect circle defined by the points inside. The blobby is the outer curve, ours is the inner curve.

Naturally the first approach that comes to mind is blobbies[Blinn 1982]. For irregularly shaped blobs containing only a few particles, this works beautifully. Unfortunately, it seems unable to match a surface such as a flat plane, a cone, or a large sphere from a large quantity of irregularly spaced particles—as we might well see in at least the initial conditions of a simulation. Bumps relating to the particle distribution are always visible. A small improvement to this was suggested in [Müller et al. 2003], where the contribution from a particle was divided by the SPH estimate of density. This overcomes some scaling issues but does not significantly reduce the bumps on what should be flat surfaces.

We thus take a different approach, guided by the principle that we should exactly reconstruct the signed distance field of an isolated particle: for a single particle x_0 with radius r_0, our implicit function must be:

$$\phi(x) = |x - x_0| - r_0 \tag{6}$$

To generalize this, we use the same formula with x_0 replaced by a weighted average of the nearby particle positions and r_0 replaced a weighted average of their radii:

$$\begin{aligned} \phi(x) &= |x - \bar{x}| - \bar{r} && (7)\\ \bar{x} &= \sum_i w_i x_i && (8)\\ \bar{r} &= \sum_i w_i r_i && (9)\\ w_i &= \frac{k(|x - x_i|/R)}{\sum_j k(|x - x_j|/R)} && (10) \end{aligned}$$

where k is a suitable kernel function that smoothly drops to zero—we used $k(s) = \max(0, (1-s^2)^3)$ since it avoids the need for square roots and is reasonably well-shaped—and where R is the radius of the neighborhood we consider around x. Typically we choose R to be twice the average particle spacing. As long as the particle radii are bounded away from zero (say at least 1/2 the particle spacing) we have found this formula gives excellent agreement with flat or smooth surfaces. See figure 4 for a comparison with blobbies using the same kernel function.

The most significant problem with this definition is artifacts in concave regions: spurious blobs of surface can appear, since $\bar{x}$ may erroneously end up outside the surface in concavities. However, since these artifacts are significantly smaller than the particle radii we can easily remove them without destroying true features by sampling $\phi(x)$ on a higher resolution grid and then doing a simple smoothing pass.

A secondary problem is that we require the radii to be accurate estimates of distance to the surface. This is nontrivial after the first frame; in the absence of a fast method of computing these to the desired precision, we currently fix all the particle radii to the constant average particle spacing and simply adjust our initial partial positions so that the surface particles are exactly this distance from the surface. A small amount of additional grid smoothing, restricted to decreasing ϕ to avoid destroying features, reduces bump artifacts at later frames.

On the other hand, we do enjoy significant advantages over reconstruction methods that are tied to level set simulations. The cost per frame is quite low—even in our unoptimized version which calculates and writes out a full 250^3 grid, we average 40–50 seconds a frame on a 2GHz G5, with the bulk of the time spent on I/O. Moreover, every frame is independent, so this is easily farmed out to multiple CPU's and machines running in parallel.

If we can eliminate the need for grid-based smoothing in the future, perhaps adapting an MLS approach such as in Shen et al.[2004], we could do the surface reconstruction on the fly during rendering. Apart from speed, the biggest advantage this would bring would be accurate motion blur for fluids. The current technique for generating intermediate surfaces between frames (for a Monte Carlo motion blur solution) is to simply interpolate between level set grid values[Enright et al. 2002b]. However, this approach destroys small features that move further than their width in one frame: the interpolated values may all be positive. Unfortunately it's exactly these small and fast-moving features that most demand motion blur. With particle-based surface reconstruction, the positions of the particles can be interpolated instead, so that features are preserved at intermediate times.

6 Examples

We used `pbrt`[Pharr and Humphreys 2004] for rendering. For the textured sand shading, we blended together a volumetric texture advected around by the simulation particles, similar to the approach of Lamorlette[2001] for fireballs.

The bunny example in figures 1 and 2 were simulated with 269,322 particles on a 100^3 grid, taking approximately 6 seconds per frame on a 2Ghz G5 workstation. We believe optimizing the particle transfer code and using BLAS routines for the linear solve would substantially improve this performance. Unoptimized smoothing and text I/O cause the surface reconstruction on a 250^3 grid to take 40–50 seconds per frame.

The column test in figures 5 and 6 was simulated with 433,479 particles on a $100 \times 60 \times 60$ grid, taking approximately 12 seconds per frame. Our cost is essentially linearly proportional to the number of particles (or equivalently, occupied grid cells).

7 Conclusion

We have presented a method for converting an existing fluid solver into one capable of plausibly animating granular materials such as sand. In addition, we developed a new fluid solver that combines the strengths of both particles and grids, offering enhanced flexibility and efficiency. We offered a new method for reconstructing implicit surfaces from particles. Looking toward the future, we plan to more aggressively exploit the optimizations available with PIC/FLIP (e.g. using multiple bounding grids), increase the accuracy of our boundary conditions, and implement motion blur of the reconstructed surface.

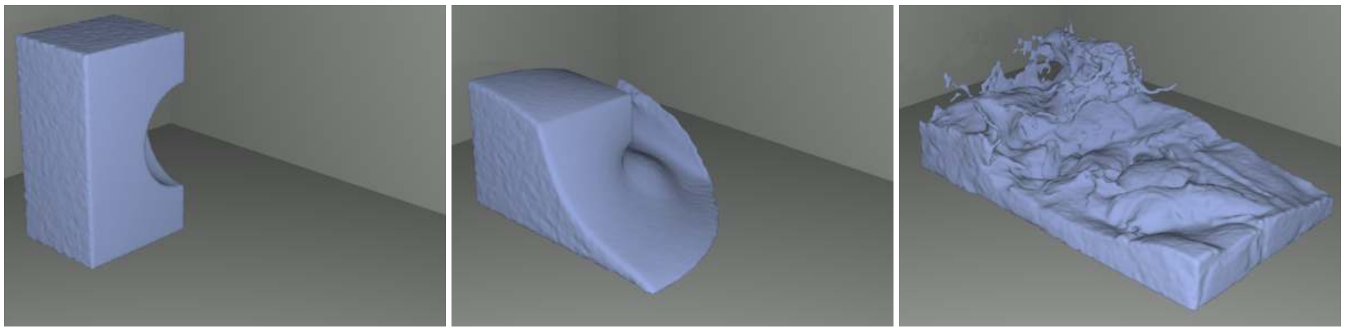
Figure 5: A column of regular liquid is released.

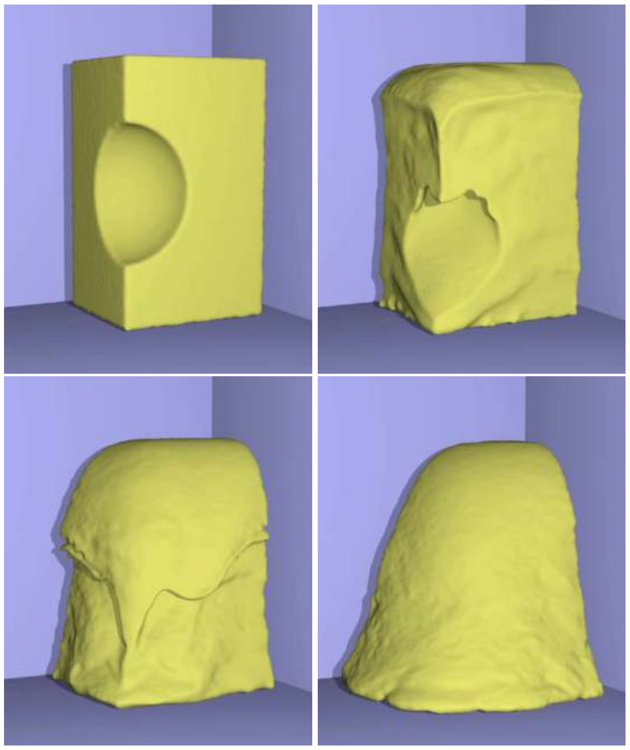
Figure 6: A column of granular material is released.

8 Acknowledgements

This work was in part supported by a grant from the Natural Sciences and Engineering Research Council of Canada. We would like to thank Dr. Dawn Shuttle for her help with granular constitutive models.

References

ADALSTEINSSON, D., AND SETHIAN, J. 1999. The fast construction of extension velocities in level set methods. *J. Comput. Phys. 148*, 2–22.

BARDENHAGEN, S. G., BRACKBILL, J. U., AND SULSKY, D. 2000. The material-point method for granular materials. *Comput. Methods Appl. Mech. Engrg. 187*, 529–541.

BLINN, J. 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3, 235–256.

BRACKBILL, J. U., AND RUPPEL, H. M. 1986. FLIP: a method for adaptively zoned, particle-in-cell calculuations of fluid flows in two dimensions. *J. Comp. Phys. 65*, 314–343.

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

CARLSON, M., MUCHA, P., VAN HORN III, R., AND TURK, G. 2002. Melting and flowing. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. 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. (Proc. SIGGRAPH) 23*, 377–384.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Comput. Anim. and Sim. '96 (Proc. of EG Workshop on Anim. and Sim.)*, Springer-Verlag, R. Boulic and G. Hegron, Eds., 61–76. Published under the name Marie-Paule Gascuel.

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

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

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

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

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

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

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 463–468.

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

HARLOW, F., AND WELCH, J. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids 8*, 2182–2189.

HARLOW, F. H. 1963. The particle-in-cell method for numerical solution of problems in fluid dynamics. In *Experimental arithmetic, high-speed computations and mathematics*.

HERRMANN, H. J., AND LUDING, S. 1998. Modeling granular media on the computer. *Continuum Mech. Therm. 10*, 189–231.

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

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 131–140.

JAEGER, H. M., NAGEL, S. R., AND BEHRINGER, R. P. 1996. Granular solids, liquids, and gases. *Rev. Mod. Phys. 68*, 4, 1259–1273.

KONAGAI, K., AND JOHANSSON, J. 2001. Two dimensional Lagrangian particle finite-difference method for modeling large soil deformations. *Structural Eng./Earthquake Eng., JSCE 18*, 2, 105s–110s.

KOTHE, D. B., AND BRACKBILL, J. U. 1992. FLIP-INC: a particle-in-cell method for incompressible flows. *Unpublished manuscript*.

LAMORLETTE, A. 2001. Shrek effects—flames and dragon fireballs. *SIGGRAPH Course Notes, Course 19*, 55–66.

LI, X., AND MOSHELL, J. M. 1993. Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *Proc. SIGGRAPH*, 361–368.

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

LUCIANI, A., HABIBI, A., AND MANZOTTI, E. 1995. A multi-scale physical model of granular materials. In *Graphics Interface*, 136–146.

MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. In *Proc. SIGGRAPH*, 37–46.

MILENKOVIC, V. J. 1996. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *Proc. SIGGRAPH*, 129–136.

MILLER, G., AND PEARCE, A. 1989. Globular dynamics: a connected particle system for animating viscous fluids. In *Comput. & Graphics*, vol. 13, 305–309.

MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys. 30*, 543–574.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Graphics Interface*.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 154–159.

NAYAK, G. C., AND ZIENKIEWICZ, O. C. 1972. Elasto-plastic stress analysis. A generalization for various constitutive relations including strain softening. *Int. J. Num. Meth. Eng. 5*, 113–135.

O'BRIEN, J., BARGTEIL, A., AND HODGINS, J. 2002. Graphical modeling of ductile fracture. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 291–294.

ONOUE, K., AND NISHITA, T. 2003. Virtual sandbox. In *Pacific Graphics*, 252–262.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.

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

SAVAGE, S. B., AND HUTTER, K. 1989. The motion of a finite mass of granular material down a rough incline. *J. Flui Mech. 199*, 177–215.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 896–904.

SMITH, I. M., AND GRIFFITHS, D. V. 1998. *Programming the Finite Element Method*. J. Wiley & Sons.

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

SULSKY, D., ZHOU, S.-J., AND SCHREYER, H. L. 1995. Application of particle-in-cell method to solid mechanics. *Comp. Phys. Comm. 87*, 236–252.

SUMNER, R. W., O'BRIEN, J. F., AND HODGINS, J. K. 1998. Animating sand, mud, and snow. In *Graphics Interface*, 125–132.

SUSSMAN, M. 2003. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comp. Phys. 187*, 110–136.

TAKAHASHI, T., FUJII, H., KUNIMATSU, A., HIWADA, K., SAITO, T., TANAKA, K., AND UEKI, H. 2003. Realistic animation of fluid with splash and foam. *Comp. Graph. Forum (Eurographics Proc.) 22*, 3, 391–400.

TAKESHITA, D., OTA, S., TAMURA, M., FUJIMOTO, T., AND CHIBA, N. 2003. Visual simulation of explosive flames. In *Pacific Graphics*, 482–486.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Proc. SIGGRAPH*, 269–278.

ZHAO, H. 2005. A fast sweeping method for Eikonal equations. *Math. Comp. 74*, 603–627.