Animating Sand as a Fluid

by

Yongning Zhu

B.Sc., Peking University, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

November 2005

Abstract

My thesis presents 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 within frictional regime with only a few small additions to account for inter-grain and boundary friction, yet with visually acceptable result.

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.

Table of Contents

List of Figures

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. Especially, I want to thank my supervisor Robert Bridson for introducing me into computer graphics area, and always encouraging me to go ahead. And I want to thank Professor Ian Mitchel who has carefully read the thesis, and proposed valued suggestions.

I would like to thank all the members in Imager lab that have been with me for these two years. Without them, I wouldn't be able to finish this work. And I would like to thank all my friends both in the University of British Columbia and back in China who have always been supporting me and caring about me.

This work was in part supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

And we would like to thank Dr. Dawn Shuttle for her help with granular constitutive models.

YONGNING ZHU

The University of British Columbia
November 2005

Chapter 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. There is an increasing interest in both physics and engineering research in studying and simulating granular materials too. Thus we are interested in animating them. People have studied animating sand using a variety of different methods. More efficient techniques still need to be developed. This thesis considers an efficient way of simulating slow flow of sand, and improvements in corresponding high quality rendering .

In this section, we will clarify the basic assumption and regime in our simulation, with a brief introduction to related works people have been doing in both computer graphics and physics. We will also give an overview of contributions of this thesis.

1.1 Basic assumption

1.1.1 Continuum Assumption

Granular material consists of so many grains that it also shows a collective behaviour. This dual property introduces special difficulties and interesting phenomena, and attracted a lot of physicists. From the numerical simulation point of view, there could be both discrete and continuous descriptions and approaches. The discrete approach, like molecular dynamics methods, is the most direct one. It captures the behaviour of every individual particle. It plays an important role for studying the macroscopic properties of granular material.

Weber, Hoffman and Hrenya[57] have performed simulations of rapid shear flow with low-to-medium volume fraction consisting of 2292 particles. They used hard-sphere collisions with cohesive forces incorporated via a square-well potential. Silbert et al[49] have done simulations of 1000-20000 particles with a soft-sphere model. Discrete models are widely used in simulation for physics.

However, this kind of approach takes too much computation as the number of particles increases. Scaling this up to just a handful of sand is clearly infeasible; a sand dune containing trillions of grains is out of the question. In this sense, discrete models aren't a good approach for animation.

From a macroscopic point of view, we cannot distinguish separate grains. Instead, we see the sand as a continuous material, behaving like a kind of fluid. More precisely, although each particle has its own velocity, we only consider their average velocity over a volume element small enough in a macroscopic view. Similarly, an averaged stress tensor will decide the behaviour of this granular material instead of particle-particle interaction. Our simulation will be based on a collective behaviour of the sand particles instead of only considering each particle. So we refer to pressure, density, viscosity , etc. in a similar sense.

Here we introduce the concept *granular flow* as a continuous description for granular material. Generally, a granular flow is a multiphase (i.e., solid and fluid) flow consisting of macroscopic solid particulates and an interstitial fluid. When sheared, the particulates may either flow in a manner similar to a fluid, or resist the shearing like a solid. This includes both dry and wet sand, but in this thesis, we only consider the flow of dry sand grains.

Therefore we will describe the sand flow with such quantities:

$\rho(x;t)$	density (kg/m^3)
$p(x;t)$	pressure (N/m^2)
$u(x;t)$	velocity (m/s)
$\sigma(x;t)$	stress tensor (N/m^2)
$f(x;t)$	body forces density(N/m^3)
$D(x;t)$	rate of strain$(1/s)$

1.1.2 Frictional flow

Granular material's behaviour could be divided into three fundamental regimes.

At low concentration and high shear rates, friction between grains can be neglected compared to collisional force. This case is called the rapid (or collisional) granular flow regime. Similar concepts and analysis from molecular kinetics are proposed. and hydrodynamic equations are derived with constitutive equations for the density, velocity and granular temperature.[19] Another mechanism, kinetic-collisional stress has been introduced for the constitutive law.[9]

When the granular flow is moving slowly, i.e. with low rate-of-strain and high concentration, it is called frictional regime, since frictional stress dominates in the interaction of grains. The flow behaves like a rigid-plastic flow. Schaeffer first introduced the incompressible granular flow model for the frictional case, and

discussed its ill-posedness in 1987[47].

Between these two cases, is the intermediate regime where both collisional and frictional inteactions must be considered.

In this thesis, we will only consider the frictional regime, and assume it to be perfect rigid-plastic flow. For simplicity, when we refer to sand behaviour in this thesis, we are mostly talking about this frictional regime.

1.2 Related works

1.2.1 Sand in Graphics

Several authors have worked on sand animation, beginning with Miller and Pearce[36]. They presented a method for animating viscous fluids by simulating the forces of such particles interacting with each other, and demonstrated a simple particle system model with short-range interaction forces which could be tuned to achieve (amongst other things) powder-like behaviour. Later Luciani et al.[33] developed a similar multi-scale 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[31] presented a dynamic height-field simulation of soil based on the standard engineering Mohr-Coulomb constitutive model of granular materials. They used a dynamic model for soil slippage and soil manipulations under the constraint of volume conservation, and developed a numerical algorithm of linear time and space complexities to meet the need of realtime computer simulation. Sumner et al.[53] took a similar heightfield approach with simple displacement and erosion rules to model footprints, tracks, etc. Onoue and Nishita[42] recently extended this to multi-valued heightfields, allowing for some 3D effects. And recently, Bell[3] simulated granular material with discrete elements represented by particles to handle difficult topology evolved.

1.2.2 Sand grains as rigid bodies

As we mentioned above, one approach to granular materials is directly simulating the grains as rigid bodies. Milenkovic[34] demonstrated a simulation of 1000 rigid spheres falling through an hour-glass using optimization methods for resolving contact. Milenkovic and Schmidl[35] improved the capability of this algorithm by using quadratic programming (QP) to increase the integration time steps, and Guendelman et al.[20] further accelerated rigid body simulation, showing 500 nonconvex rigid bodies falling through a funnel.

1.2.3 Particle-based Water Simulation

Particles have been a popular technique in graphics for water simulation. Smoothed Particle Hydrodynamics(SPH) was introduced for computational fluid dynamics, by representing the fluid with a set of particles. These particles work as matter elements or sample points of the values or derivatives of physics quantities. SPH has an obvious advantage that pure advection is treated exactly in SPH, while for finite difference schemes, numerical error will smooth out details in advections. The SPH method has been applied widely including diffusion problems, turbulence and multiphase flow problems. See Monaghan[37] for the standard review of SPH.

Desbrun and Cani[10] introduced SPH to the animation literature, demonstrating a viscous fluid flow simulation using particles alone.

Müller et al.[39] developed SPH further for water simulation, achieving impressive interactive results. Premoze et al.[44] 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.

1.2.4 Grid-based Water Simulation

Particle-based simulation can suffer from high computational costs and stability problems. There has been more work on animating water with grids for efficient simulation with better stability. Foster and Metaxas[15] developed the first grid-based fully 3D water simulation in graphics. They were using a staggered grid for solving the Navier-Stokes equation, and marker particles to track the fluid position. Velocity advection is treated with forward integration. Stam[51] provided the semi-Lagrangian advection method which is unconditionally stable, and thus allows time steps large enough for interactive animation. The excessive numerical dissipation in that work was later mitigated by Fedkiw et al[14] with higher order interpolation and vorticity confinement.

One of the problems in grid-based algorithms is volume loss. To avoid volume loss for pure level set advection and the poor artifacts of the surface directly generated from particles, Foster and Fedkiw[16] incorporated former work into a water solver, where particles are combined with a level set function to express the fluid distribution. To this model Génevaux et al.[17] added two-way fluid-solid interaction forces. The interaction between fluid and solid is introduced by a damped spring between each marker particle near solid object and each nodal mass of the solid. Carlson et al.[7] added variable viscosity to a water solver, providing beautiful simulations of wet sand dripping, which is achieved simply by increased viscosity. Enright et al.[13] extended the Foster and Fedkiw approach with particles on both sides of the interface and velocity extrapolation into the air. Losasso et al.[32] extended this to dynamically adapted octree grids, providing much enhanced resolution, and Rasmussen et al.[45] improved the boundary conditions and velocity extrapolation while adding several other features important for visual effects production. Carlson et al.[8] added better coupling between fluid and rigid body simulations—which we

parenthetically note had its origins in scientific work on fluid flow with sand grains. Hong and Kim[25] and Takahashi et al.[55] 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).

1.2.5 Plastic Flow

For more general plastic flow, most of the graphics work has dealt with meshes. Terzopoulos and Fleischer[56] first introduced plasticity to physics-based animation, with a simple 1D model applied to their springs. O'Brien et al.[41] added plasticity to a fracture-capable tetrahedral-mesh finite element simulation to support ductile fracture. Irving et al.[26] 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. [38]). Closest in spirit to the current paper, Goktekin et al.[18] added elastic forces and associated plastic flow to a water solver, enabling animation of a wide variety of non-Newtonian fluids.

1.2.6 Scientific Work on Granular Materials

For a scientific description of the physics of granular materials, we refer the reader to Jaeger et al.[27]. In the soil mechanics literature there is a long history of numerically simulating granular materials using an 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[40], and the book by Smith and Griffiths[50] 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[46]. For alternatives

that simulate the material at the level of individual grains, Herrmann and Luding's review[24] is a good place to start.

1.2.7 Scientific Work on Fluid Flow

Our new fluid code traces its history back to the early particle-in-cell (PIC) work of Harlow[23] for compressible flow. Shortly thereafter Harlow and Welch[22] 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[5]; Kothe and Brackbill[29] later worked on adapting FLIP to incompressible flow. Compressible FLIP was also extended to an elasto-plastic finite element formulation, the Material Point Method[52], which has been used to model granular materials at the level of individual grains[2] amongst other things. MPM was used by Konagai and Johansson[28] for simulating large-scale (continuum) granular materials, though only in 2D and at considerable computational expense.

1.3 Overview of the Thesis

In Chapter 2, we explain our sand modelling. Our constitutive law is based on a continuous description about granular flow, and assuming our frictional sand flow to be perfectly rigid-plastic. Frictional boundary conditions are applied and details about the numerical implementation is discussed.

In Chapter 3, we also present a new fluid simulation method. 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 including boundary conditions, incompressibility, and in the case of sand, friction forces.

Our simulations only output particles that indicate where the fluid (or sand flow) is. To actually render the result, we need to reconstruct a smooth surface that wraps around these particles. In Chapter 4 we explain our improvement for the level set reconstruction, and the advantages this framework has over existing approaches.

Chapter 2

Sand Modeling

As we assumed, we consider sand as a kind of continuous material, and apply fluid dynamics expressions to describe the behavior of sand. In this chapter, we will first review the general fluid dynamics concepts and description of fluid simulation to clarify concepts.

The dynamics of granular materials have been studied extensively in the field of soil mechanics. There have been over one hundred kinds of different models about constitutive relationship of granular flow. We will introduce our modeling of sand, or constitutive law, assuming that sand flow is perfectly rigid-plastic with the Mohr-Coulomb plasticity law. We will discusfs in detail about frictional stress which makes sand different from real fluid.

To achieve efficient simulations for animation purposes, we made some simplified assumptions for the sand modeling. These assumptions allow us to express our sand simulation as an extension of a fluid simulation process. The conditions for these assumptions as well as their advantages and disadvantages will also be clarified. Details of our sand simulation algorithm, as well as some implementation issues will also be discussed in the last part.

2.1 Fluid dynamics

In the field of hydrodynamics, there are two ways to describe fluid properties. Lagrangian framework is more natural from kinetic theory of molecules. In the Lagrangian framework, a fluid is considered to be a collection of a lot of abstract moving particles, each particle's behavior is described by the state and position as functions of time. In many cases, the initial position of that particle is used for reference.

In the Eulerian approach, all fluid properties, including velocity and pressure, are considered as a function of space and time. For any physics property in Table 1.1.1,the material time derivative D/Dt measures how the quantity at a point in space changes as the material flows past as well as changes. For example, using the chain rule with the density ρ illustrates the material time derivative:

$$\frac{D}{Dt}\rho(t;p) = \frac{\partial}{\partial t}\rho(x(t;p),t) + \frac{d}{dt}x(t;p)\cdot\nabla\rho(x(t;p),t) = \rho(x,t)_t + u(x,t)\cdot\nabla\rho(x,t).$$

This also explains derivative relationship between Eulerian and Lagrangian description

Incompressible fluid

For incompressible fluids, the density remains constant in time as it is advected with the flow, i.e. the material derivative of density is zero:

$$\frac{D}{Dt}\rho(t;p) = \rho(x,t)_t + u(x,t)\cdot\nabla\rho(x,t) = 0 \ \forall p \text{ s.t. } x(0;p)\in\Omega(0).$$

In this thesis, we will only consider the simple case of ρ being constant, which is a sufficient but not necessary condition for incompressibility.

Mass conservation and Momentum conservation Mass conservation and momentum conservation can easily be derived by taking the time derivative of

the mass or momentum integrated over a volume element(see [21] for example):

$$\rho_t + (u \cdot \nabla)\rho = 0 \tag{2.1}$$

$$\rho u_t + u \cdot \nabla(\rho u) = \rho f + \nabla \cdot \sigma \tag{2.2}$$

where σ is the stress tensor. In a Cartesian coordinate system, we know that the three columns of σ are actually interaction tractions over the three coordinate planes of a cubic volume element. σ must always be a symmetric tensor, so $\sigma_{i,j}$ is actually the stress on the i-plane in j direction.

Conventionally, the isotropic part of sigma is separated from the rest of the stress tensor, corresponding to the fluid pressure p:

$$\sigma = \tau - pI$$

Navier-Stokes Equation

For Newtonian fluids, like water and thin mineral oils,

$$\tau = \eta \left(\frac{\nabla u + \nabla u^T}{2} \right)$$

where η is the dynamic viscosity, $\eta = \rho\nu$. In this thesis, we ignore difference in the temperature for the fluids, and take viscosity ν to be constant, so

$$\rho u_t + u \cdot \nabla(\rho u) = \rho f + \eta \nabla \cdot \nabla u - \nabla \cdot pI$$

$$u_t + u \cdot \nabla u = f + \nu \nabla \cdot \nabla u - \frac{\nabla p}{\rho}$$

This equation, in combination with mass conservation equation for incompressible fluid forms the basic Navier-Stokes equation :

$$\begin{cases} \nabla \cdot u &= 0 \\ u_t + u \cdot \nabla u &= f + \nu \nabla \cdot \nabla u - \frac{\nabla p}{\rho} \end{cases}$$

Towards sand

Newtonian flow is quite different from plastic flow which occurs only when a finite minimum force is exceeded. Sand is usually modeled as elasto-plastic flow or rigid-plastic flow. Then the major problem is how to decide stress σ, or rather the shearing part of it τ.

2.2 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

$$\sqrt{3}\bar{\sigma} < \sin\phi \sigma_m \tag{2.3}$$

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 $\sin\phi$.

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[41, 18, 26]), they are impossible here. The Mohr-Coulomb law compares the shear part of the

[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.

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.

2.3 A Simplified Model

2.3.1 Assumptions for simplification

We first ignore the nearly imperceptible elastic deformation regime when there is no yielding, as the elasticity in rock is so small. And we also ignore the tiny volume changes that sand undergoes as it starts and stops flowing. This is correct only for the frictional flow regime which is what we are concentrating on. 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.

A more accurate model was given by Schaeffer[47]. They discussed a rigid-perfectly plastic, incompressible, cohesionless Coulomb granular flow. But they didn't make the pressure assumption, instead, they solve pressure directly. Since pressure appears not only in making the velocity incompressible, but also in stress tensor term, this will introduce a lot more work in Finite Element solving.

2.3.2 Mohr-Coulomb law

The constitutive law has two parts: the yield condition and the flow rule.

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

$$\sigma_f = -\sin\phi p \frac{D}{\sqrt{1/3}|D|_F} \tag{2.4}$$

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. Although this is frictional stress only for yielding region, we compute it for all fluid cells for yield condition. In fact , this value recognizes the maximum frictional force under current normal pressure to resist relative movement between grid cells . The real frictional stress could be either equal to that on yielding or smaller than that when smaller static friction could afford the resisting of yielding.

[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.

In fact, since the stress tensor has a linear relationship with the direction of rate of strain instead of rate of strain itself, such modification might be affecting the incompressibility in velocity. By the numerical result, such incompressibility is visually acceptable.

We finally need a way of determining the yield condition without tracking elastic stress and strain, i.e. when the sand should be rigidified. 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:

$$\sigma_{rigid} = -\frac{\rho D \Delta x^2}{\Delta t} \tag{2.5}$$

This is actually the minimum stress required for rigidifying sand between two grid cells. And we can compare this with the maximum frictional stress that could be afforded corresponding to current normal pressure σ_f, and check if this satisfies our yield inequality: if it does, i.e. if the required friction for resisting movement is small enough to be afforded with current pressure, and hence the sand can resist forces and inertia trying to make it slide, then it could be kept static with a static friction, and the material in that cell should be rigid at the end of the time step. Otherwise, when the required friction for resisting movement is bigger than maximum friction stress, then the real friction stress is just this maximum friction stress. And the sand starts to slide.

2.3.3 Algorithm

This gives us the following algorithm for sand simulation.

- For every time
  - Apply advection
  - Apply gravity and boundary conditions, solve for pressure to enforce incompressibility, and subtracting $\Delta t/\rho \nabla p$ from the intermediate velocity field to make it incompressible.
  - Evaluate the strain rate tensor D in each grid cell with standard central differences, and decide yielding condition for each grid cell:
    - If the resulting σ_{rigid} from equation 2.5 satisfies the yield condition with 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 2.4.
  - Treat the two regions differently:
    - Find all connected groups of rigid cells, and project the velocity field in each separate group to the space of rigid body motions.
    - For all the yielding cells, we update with the frictional stress, using standard central differences for $u+ = \Delta t/\rho \nabla \cdot \sigma_f$.
- EndFor

2.3.4 Sand rigidification

For rigid cells, we will project the velocity field in each separate group to the space of a corresponding rigid body movement velocity. Similar to the rigid fluid work by Carlson et. al.[7], recall that rigid body motion is determined entirely by the mass center velocity $\bar{\mathbf{v}}$ and angular velocity $\bar{\omega}$. These can be easily calculated via velocity

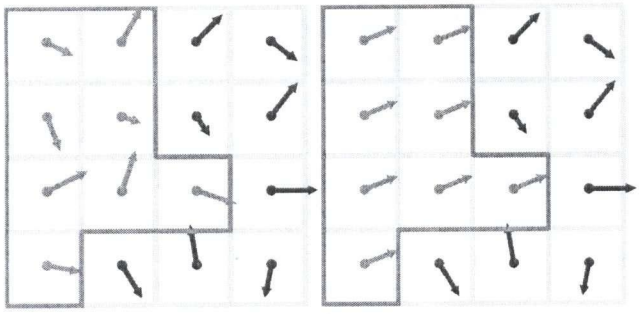

Figure 2.1: Solidification of non-yielding grid cells.

and angular momentum conservation(Fig 2.1).

$$\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} m_i \mathbf{v_i} = \Big(\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} m_i\Big)\bar{\mathbf{v}} \tag{2.6}$$

$$\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} m_i \mathbf{r_i} \times \mathbf{v_i} = \Big(\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} m_i\Big)\bar{\omega} \tag{2.7}$$

where $\mathbf{r}$ is the position of each cell. The projected rigid velocity in this region is

$$\mathbf{v}(\mathbf{x}) = \bar{\mathbf{v}} + (\mathbf{x} - \bar{\mathbf{x}}) \times \bar{\omega}$$

where

$$\bar{\mathbf{x}} = \Big(\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} m_i \mathbf{x_i}\Big) / \Big(\sum_{\substack{\text{grid cells to}\\ \text{be rigidified}}} \mathbf{m_i}\Big)$$

2.4 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.[6] 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,

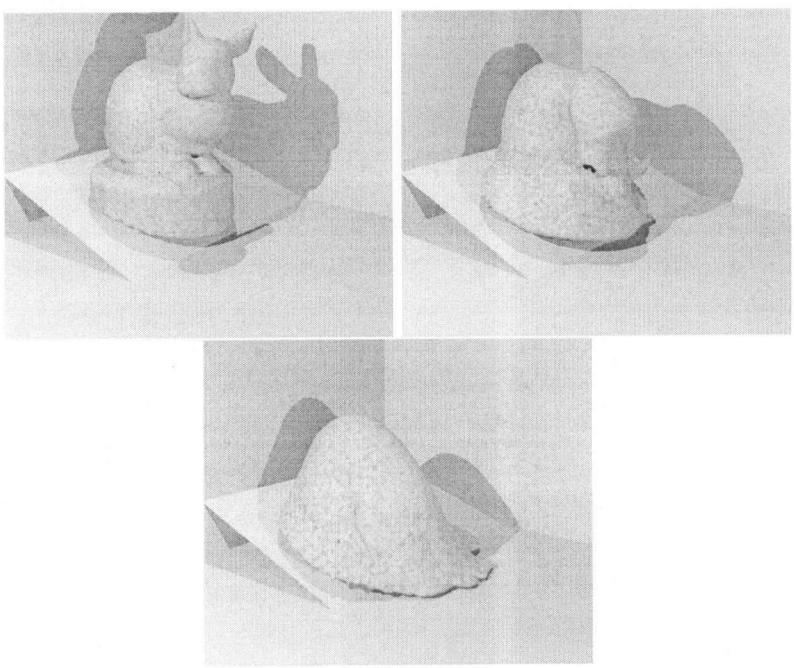

Figure 2.2: The sand bunny with a boundary friction coefficient $\mu = 0.6$: compare to figure 5.1 where zero boundary friction was used.

clamping it to zero for the case of static friction:

$$u_T = \max\left(0, 1 - \frac{\mu|u \cdot n|}{|u_T|}\right) u_T \tag{2.8}$$

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 2.2, which includes friction, to figure 5.1 which doesn't.

2.5 Cohesion

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

$$\sqrt{3}\bar{\sigma} < \sin\phi\sigma_m + c \tag{2.9}$$

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

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.

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. However, for modeling sand with low cohesion, introducing a little cohesion will help in stablization.

Chapter 3

Fluid Simulation Revisited

3.1 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 [8, 18, 32]) 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 [37, 39, 44] 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[11] 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[54] 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[16] and later Enright et al.[13] 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[32], 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.

3.2 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[23]. 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[5] 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[1] as follows:

- Initialize particle positions and velocities(Fig 3.1(a))
- For each time step:
  - At each staggered MAC grid node, compute a weighted average of the nearby particle velocities(Fig 3.1(b))
  - For FLIP: Save the grid velocities.

[1]The one precedent for this that we could find is a reference to an unpublished manuscript[29].

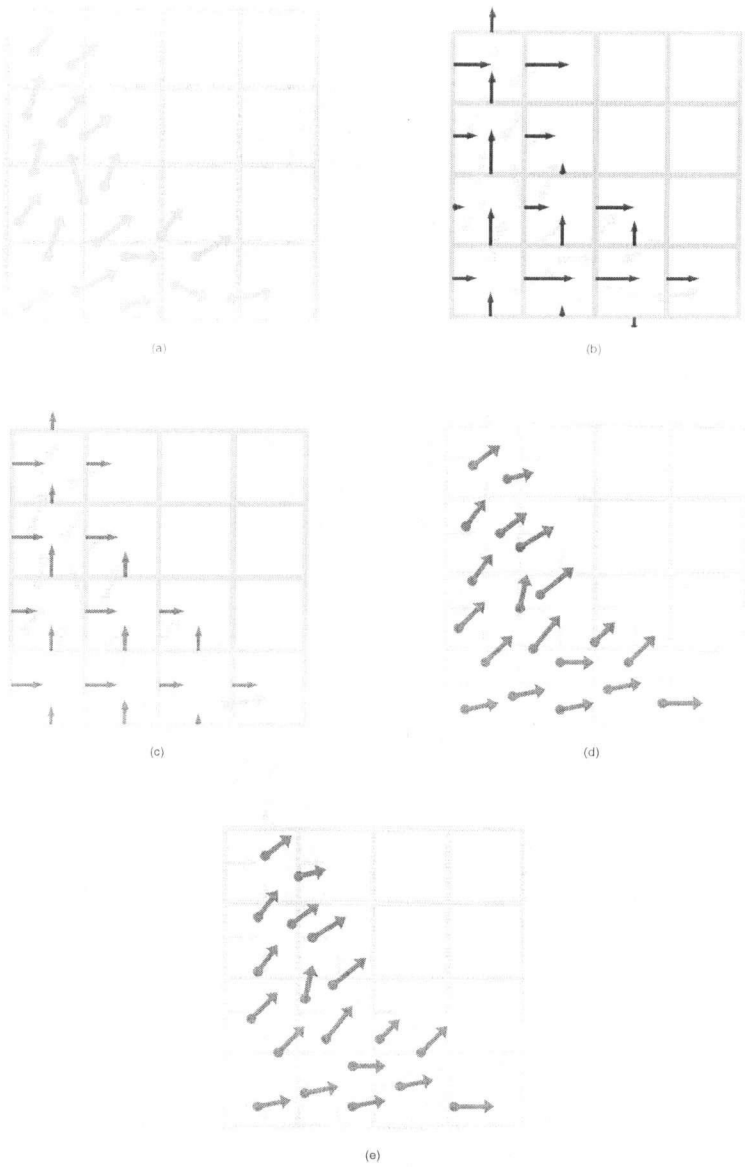

Figure 3.1: PIC and FLIP Method. (a) Algorithm starts with particles carrying their velocities. (b) Transfer velocities from particles to grid.(c) Resolve forces on grid, including external forces like gravity, resolving boundary conditions and projecting velocity back to be incompressible.(d) Update particle velocities.(e) Update particle position carrying velocities with them(advection).

- Do all the non-advection steps of a standard water simulator on the grid.(Fig 3.1(c), 3.1(d))
- 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.(Fig 3.1(e))
- 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.

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 fewer particles per grid cell, we tend to run into occasional "gaps" in the flow, and more particles will be introducing much more computation.

To help with surface reconstruction later (section 4) 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.

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

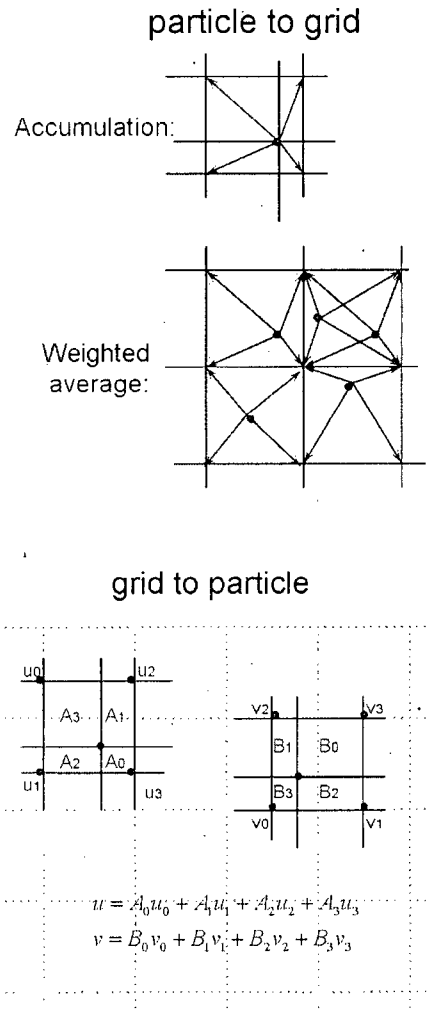

Figure 3.2: Velocity interpolation between particles and grid cells.

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(Fig 3.2). 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[12] 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.

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[58] 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.[13], 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[1]).

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(Fig 3.2). For viscous

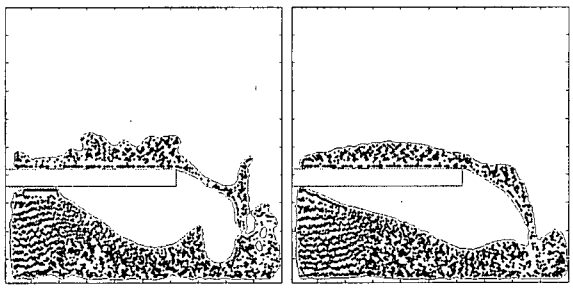

Figure 3.3: FLIP vs. PIC velocity update for the same simulation. Notice the small-scale velocities preserved by FLIP but smoothed away by PIC.

flows, such as sand, the additional numerical viscosity of PIC is beneficial. In the PIC method, the velocity is smoothed out by constantly averaging and interpolating at each time step which introduces obvious numerical viscosity. For inviscid flows, such as water, FLIP is preferable. In fact, in the FLIP method, only velocity increment is interpolated once for each time step. See figure 3.3 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. For example, a viscosity step on the grid in concert with PIC would be required for highly viscous fluids.

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.[13]. 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. This will introduce loss of incompressibility, hence, we used better pressure solver and more accurate boundary condition to avoid it.

3.2.1 Error analysis

Error is introduced during the process of transfering velocity back and forth between grid and particles.For a simple analysis, we consider the 1-dimensional case, and consider the value of a function f within two grid cells: $[x_{-1}, x_0], [x_0, x_1]$.

Transfer from grid to particles

When we compute f on certain position x between x_0 and x_1, the interpolated value $\hat{f}(x)$ could be approximated by Taylor expansion:

$$f(x_0) = f(x) + (x_0 - x)f'(x) + \frac{(x_0 - x)^2}{2}f''(x) \tag{3.1}$$

$$f(x_1) = f(x) + (x_1 - x)f'(x) + \frac{(x_1 - x)^2}{2}f''(x) \tag{3.2}$$

$$\hat{f}(x) \equiv (1 - t)f(x_0) + tf(x_1) \tag{3.3}$$

$$\text{with } t = \frac{x - x_0}{x_1 - x_0}, 1 - t = \frac{x_1 - x}{x_1 - x_0} \tag{3.4}$$

$$\therefore \hat{f}(x) = f(x) - \frac{f''(x)}{2}(x_0 - x)(x_1 - x) \tag{3.5}$$

Transfer from particles to grid

Without loss of generality, we assume $x_1 - x_0 = x_0 - x_{-1} = \Delta x$. And we assume that the particles appear with uniform probability over each section, with interpolation error from 3.1, then the expectation of new value of f, $\tilde{f}(x)$ transferred from particles' values back to grid is

$$\tilde{f}(x_0) = \frac{\int_{\frac{x_{-1}+x_0}{2}}^{\frac{x_0+x_1}{2}} w(x)\hat{f}(x)\,dx}{\int_{\frac{x_{-1}+x_0}{2}}^{\frac{x_0+x_1}{2}} w(x)\,dx} \tag{3.6}$$

$$w(x) = \begin{cases} x - x_{-1} & x < x_0 \\ x_1 - x & x > x_0 \end{cases} \tag{3.7}$$

$$\therefore \tilde{f}(x_0) = f(x_0) + \frac{1}{9}f''(x_0)\Delta x^2 + O(\Delta x^3) \tag{3.8}$$

So ignoring advection case the error introduced by interpolating to particles and then averaged back to grid is $O(\Delta x^2)$.

Chapter 4

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[44]. While this is an effective solution, we believe a fully particle-based reconstruction can have significant advantages.

Naturally the first approach that comes to mind is blobbies[4]. 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 [39], 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

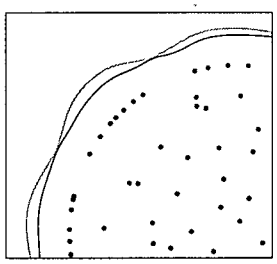

Figure 4.1: Comparison of our implicit function and blobbies for matching a perfect circle defined by the points inside. The blobby is the outer curve, ours is the inner curve.

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:

$$\phi(x) = |x - x_0| - r_0 \tag{4.1}$$

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:

$$\begin{aligned} \phi(x) &= |x - \bar{x}| - \bar{r} && (4.2) \\ \bar{x} &= \sum_i w_i x_i && (4.3) \\ \bar{r} &= \sum_i w_i r_i && (4.4) \\ w_i &= \frac{k(|x - x_i|/R)}{\sum_j k(|x - x_j|/R)} && (4.5) \end{aligned}$$

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 4.1 for a comparison with blobbies using the same kernel function.

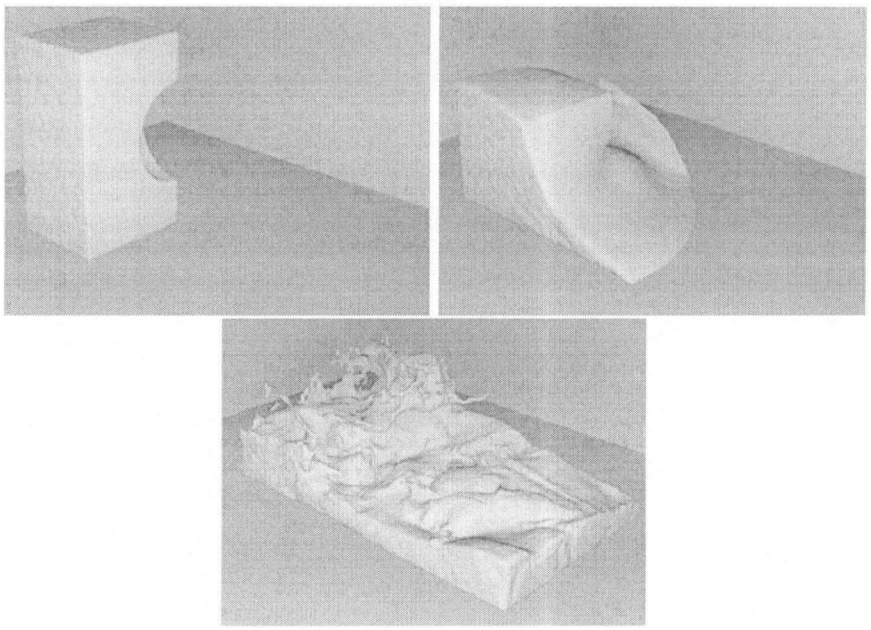

Figure 4.2: A column of regular liquid is released.

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 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

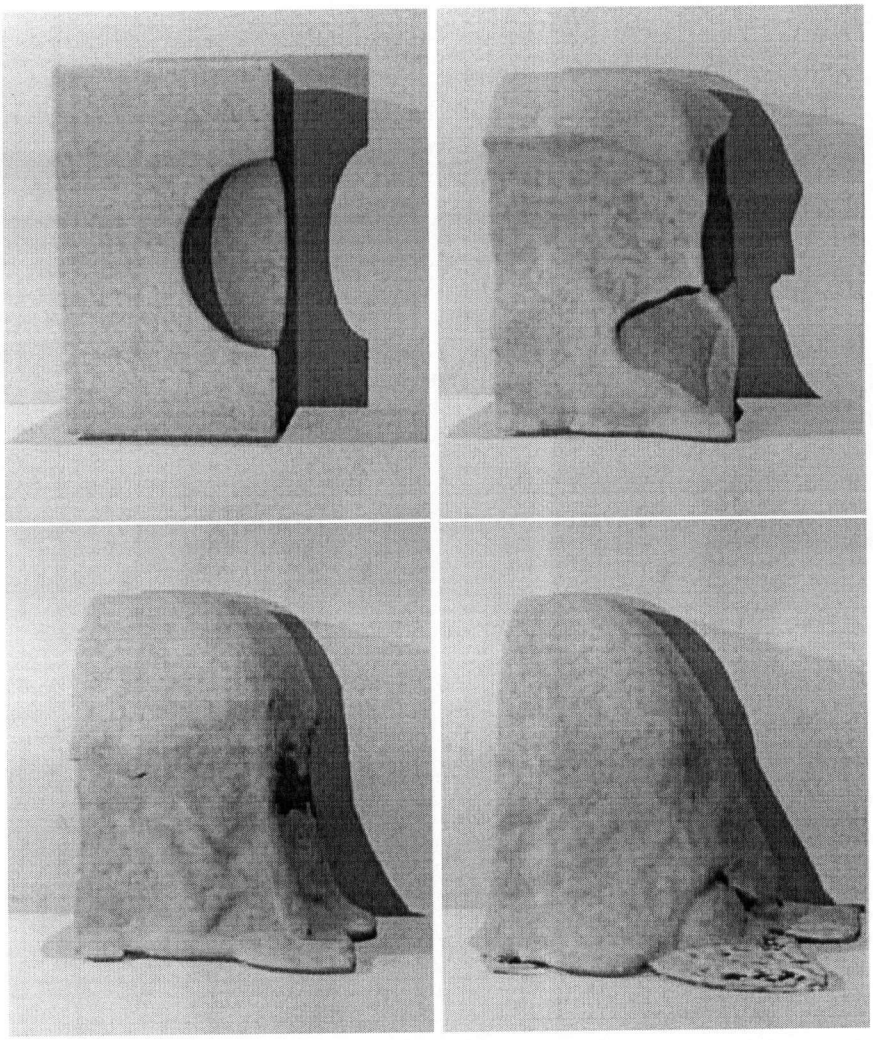

Figure 4.3: A column of sand is released.

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.[48], 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[13]. 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.

Chapter 5

Results

We used `pbrt`[43] for rendering. For the matte sand shading, we blended together a volumetric texture advected around by the simulation particles, similar to the approach of Lamorlette[30] for fireballs.

The bunny example in figures 5.1 and 2.2 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 4.3 and 4 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).

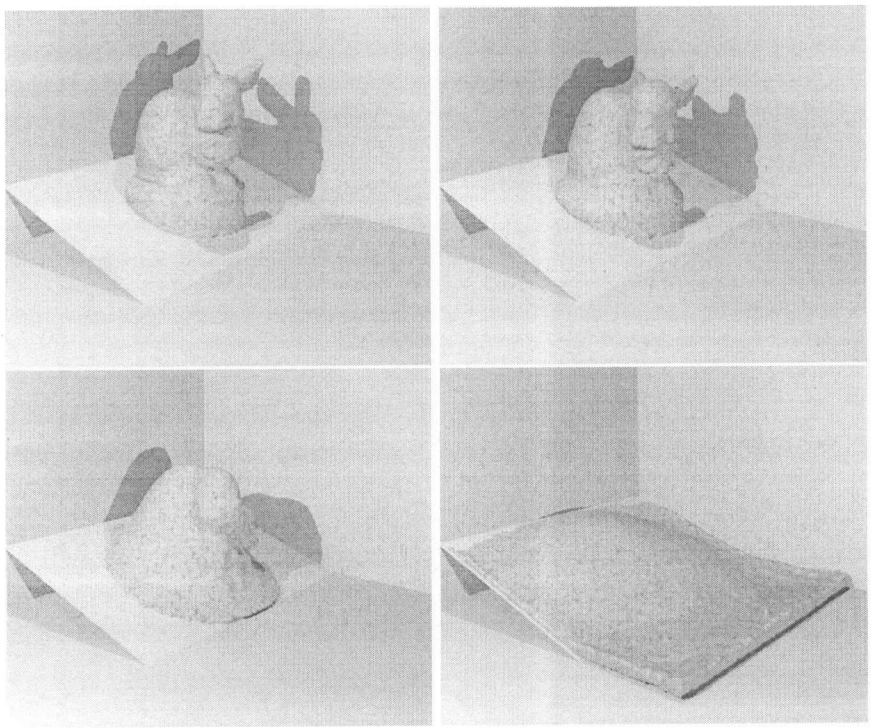

Figure 5.1: Low friction sand bunny.

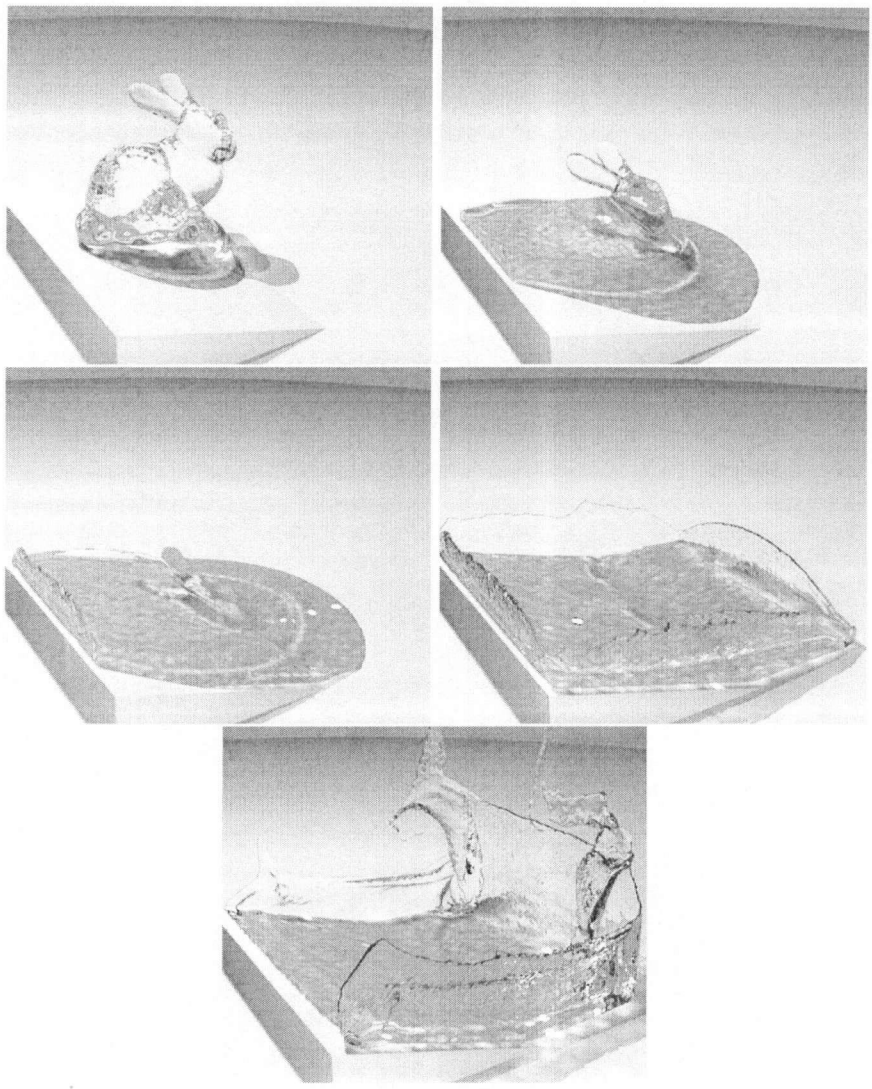

Figure 5.2: Water bunny.

Chapter 6

Future Work and Conclusions

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 implemented a new fluid solver that combines the strengths of both particles and grids, offering enhanced flexibility and efficiency. We developed a new idea 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.

Bibliography

[1] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148:2–22, 1999.

[2] S. G. Bardenhagen, J. U. Brackbill, and D. Sulsky. The material-point method for granular materials. *Comput. Methods Appl. Mech. Engrg.*, 187:529–541, 2000.

[3] Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86, New York, NY, USA, 2005. ACM Press.

[4] J. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[5] J. U. Brackbill and H. M. Ruppel. FLIP: a method for adaptively zoned, particle-in-cell calculuations of fluid flows in two dimensions. *J. Comp. Phys.*, 65:314–343, 1986.

[6] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21:594–603, 2002.

[7] Mark Carlson, Peter Mucha, R Van Horn III, and Greg Turk. Melting and flowing. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pages 167–174, 2002.

[8] Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23:377–384, 2004.

[9] Sebastien Dartevelle. *Numerical and granulometric approaches to geophysical granular flows.* PhD thesis, Michigan Technological University, 2003.

[10] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In R. Boulic and G. Hegron, editors, *Comput.*

Anim. and Sim. '96 (Proc. of EG Workshop on Anim. and Sim.), pages 61–76. Springer-Verlag, Aug 1996. Published under the name Marie-Paule Gascuel.

[11] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *J. Comp. Phys.*, 183:83–116, 2002.

[12] Doug Enright, Duc Nguyen, Frederic Gibou, and Ron Fedkiw. 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.*, number FEDSM2003–45144. ASME, 2003.

[13] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):736–744, 2002.

[14] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 15–22, 2001.

[15] N. Foster and D. Metaxas. Realistic animation of liquids. *Graph. Models and Image Processing*, 58:471–483, 1996.

[16] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 23–30, 2001.

[17] Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. Simulating fluid-solid interaction. In *Graphics Interface*, pages 31–38, 2003.

[18] Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. A method for animating viscoelastic fluids. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23:463–468, 2004.

[19] I. Goldhirsch. Rapid granular flows. *Annu. Rev. Fluid Mech.*, 35, 2003.

[20] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3):871–878, 2003.

[21] W. Malalasekra H. Versteeg. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method Approach.* Prentice Hall, 1996.

[22] F. Harlow and J. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids*, 8:2182–2189, 1965.

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

[24] H. J. Herrmann and S. Luding. Modeling granular media on the computer. *Continuum Mech. Therm.*, 10:189–231, 1998.

[25] Jeong-Mo Hong and Chang-Hun Kim. Animation of bubbles in liquid. *Comp. Graph. Forum (Eurographics Proc.)*, 22(3):253–262, 2003.

[26] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pages 131–140, 2004.

[27] H. M. Jaeger, S. R. Nagel, and R. P. Behringer. Granular solids, liquids, and gases. *Rev. Mod. Phys.*, 68(4):1259–1273, 1996.

[28] Kazuo Konagai and J. Johansson. Two dimensional Lagrangian particle finite-difference method for modeling large soil deformations. *Structural Eng./Earthquake Eng., JSCE*, 18(2):105s–110s, 2001.

[29] D. B. Kothe and J. U. Brackbill. FLIP-INC: a particle-in-cell method for incompressible flows. *Unpublished manuscript*, 1992.

[30] Arnauld Lamorlette. Shrek effects—flames and dragon fireballs. *SIGGRAPH Course Notes, Course 19*, pages 55–66, 2001.

[31] Xin Li and J. Michael Moshell. Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 361–368, 1993.

[32] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23:457–462, 2004.

[33] A. Luciani, A. Habibi, and E. Manzotti. A multi-scale physical model of granular materials. In *Graphics Interface*, pages 136–146, 1995.

[34] Victor J. Milenkovic. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 129–136, 1996.

[35] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 37–46, 2001.

[36] G. Miller and A. Pearce. Globular dynamics: a connected particle system for animating viscous fluids. In *Comput. & Graphics*, volume 13, pages 305–309, 1989.

[37] J. J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[38] M. Müller and M. Gross. Interactive virtual materials. In *Graphics Interface*, 2004.

[39] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pages 154–159, 2003.

[40] G. C. Nayak and O. C. Zienkiewicz. Elasto-plastic stress analysis. A generalization for various constitutive relations including strain softening. *Int. J. Num. Meth. Eng.*, 5:113–135, 1972.

[41] J. O'Brien, A. Bargteil, and J. Hodgins. Graphical modeling of ductile fracture. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21:291–294, 2002.

[42] K. Onoue and T. Nishita. Virtual sandbox. In *Pacific Graphics*, pages 252–262, 2003.

[43] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation.* Morgan Kaufmann, 2004.

[44] Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross Whitaker. Particle–based simulation of fluids. In *Comp. Graph. Forum (Eurographics Proc.)*, volume 22, pages 401–410, 2003.

[45] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directible photorealistic liquids. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pages 193–202, 2004.

[46] S. B. Savage and K. Hutter. The motion of a finite mass of granular material down a rough incline. *J. Fluid Mech.*, 199:177–215, 1989.

[47] D. G. Schaeffer. Instability in the evolution equations describing incompressible granular flow. *Journal of Differential Equations*, 66, 1987.

[48] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23:896–904, 2004.

[49] L.E. Silbert, D. Ertas, G. S. Grest, and et. al. Granular flow down an inclined plane: Bagnold scalling and rheology. *Physical Review E*, 64, 2001.

[50] I. M. Smith and D. V. Griffiths. *Programming the Finite Element Method.* J. Wiley & Sons, 1998.

[51] Jos Stam. Stable fluids. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 121–128, 1999.

[52] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of particle-in-cell method to solid mechanics. *Comp. Phys. Comm.*, 87:236–252, 1995.

[53] R. W. Sumner, J. F. O'Brien, and J. K. Hodgins. Animating sand, mud, and snow. In *Graphics Interface*, pages 125–132, 1998.

[54] Mark Sussman. 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, 2003.

[55] Tsunemi Takahashi, Hiroko Fujii, Atsushu Kunimatsu, Kazuhiro Hiwada, Takahiro Saito, Ken Tanaka, and Heihachi Ueki. Realistic animation of fluid with splash and foam. *Comp. Graph. Forum (Eurographics Proc.)*, 22(3):391–400, 2003.

[56] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 269–278, 1988.

[57] M. W. Weber, D. K. Hoffman, and C. M. Hrenya. Discrete-particle simulations of cohesive granular flow using a square-well potential. *Granular Matter*, 6, 2004.

[58] Hongkai Zhao. A fast sweeping method for Eikonal equations. *Math. Comp.*, 74:603–627, 2005.