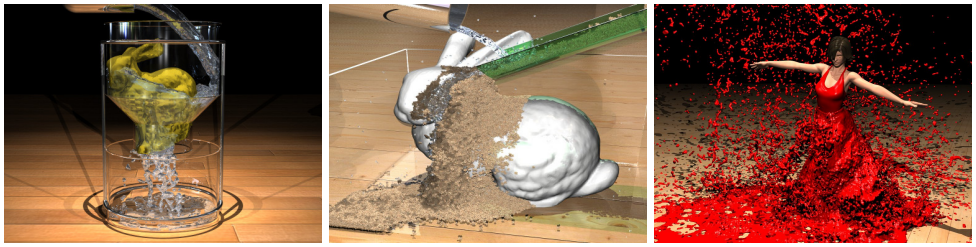
KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Computer Science



UNIFIED PARTICLE SIMULATIONS AND INTERACTIONS IN COMPUTER ANIMATION

Toon LENAERTS

Dissertation presented in
partial fulfillment of the
requirements for the degree
of Doctor in Engineering

December 2009

UNIFIED PARTICLE SIMULATIONS AND INTERACTIONS IN COMPUTER ANIMATION

Toon LENAERTS

Jury:
Prof. dr. Etienne Aernoudt, president
Prof. dr. ir. Philip Dutré, promotor
Prof. dr. ir. Dirk Roose
Prof. dr. ir. Marc Van Barel
Prof. dr. ir. Stefan Vandewalle
Prof. dr. ir. Luc Van Eycken
Prof. dr. Philippe Bekaert (Universiteit Hasselt)

Dissertation presented in
partial fulfillment of the
requirements for the degree
of Doctor in Engineering

UDC 681.3*I3

December 2009

D/2009/7515/142
ISBN 978–94–6018–162-7

Onderzoek gefinancierd met een specialisatiebeurs van het Instituut voor de Aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen (IWT-Vlaanderen).

Research funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

Preface

Before I started my Ph.D. study, I heard an older Ph.D. student say you do a Ph.D on your own and you do it all the time, even in the shower. Now, I must say I indeed was thinking particles all the time, even in the shower. However, I did not do it all alone, I was supported by many people.

Looking back to how it all started, I definitely should thank my supervisor prof. dr. ir. Philip Dutré. First, for teaching me computer graphics. Secondly, for giving me the opportunity to start a Ph.D. and for giving me the freedom of choosing topics I like the most. I also wish to thank the IWT for providing me a Ph.D. grant. In this context special thanks also go to Philippe Bekaert, one of the members of my IWT jury. Thank you for letting me convince you. I also thank my assessors, prof. dr. ir. Dirk Roose and prof. dr. ir. Marc Van Barel, and the members of the examination committee, prof. dr. Philippe Bekaert, prof. dr. ir. Stefan Vandewalle and prof. dr. ir. Luc Van Eycken for carefully reading this dissertation and their insightful comments. I thank prof. dr. ir. Etienne Arnoudt for chairing my committee

Right from the start, I was lucky to have a colleague like Bart Adams, who has a lot of experience in particle-based simulations. Bart, I thank you for bringing me up to speed, I very much enjoyed working with you and appreciate your help on our SIGGRAPH paper. I also enjoyed working and taking coffee breaks with my other colleagues, present and former: Ares Lagae, Jurgen Laurijssen, Benedict Brown, Roeland Schoukens and Peter Vangorp, Olivier Dumont, Karl vom Berge, Muath Sabha and Pieter Peers.

I thank my friends and family for their support, for being enthusiastic about my results and for all relaxing and exciting moments in pubs, at parties and everywhere else.

Looking further back to how it started, I should really thank my parents, Linda Denruyter and Paul Lenaerts. There's no measure for their support and the opportunities they gave me. Thank you.

I thank one last person, the one who always stands right next to me, the one who raises me up, the one who I can lean on to, my Carolyne.

Toon Lenaerts
Heverlee, December 2009

Abstract

Today's computer animations and movies with special effects have reached a high level of realism and complexity. Modeling and animation software has become more and more versatile and offers state of the art algorithms mimicking our real world. Amongst those algorithms are the physics-based methods for solving the dynamics of real-life phenomena.

This dissertation covers such algorithms for the simulation of fluid behavior and the dynamics of objects. We specifically concentrate on the mutual interactions between fluids and objects. Generally previous systems only simulate the geometry of objects in combination with fluids, therefore treating the object as an impenetrable solid. Yet most objects are made out of porous materials such as sponges or cloth, meaning they absorb and diffuse fluid through their body upon interaction, which affects their physical behavior.

In this work we present a novel simulation algorithm for porous flow through a wide variety of fluid-absorbent deformable objects and granular volumes. We introduce the physical principles governing porous flow, expressed by the law of Darcy, into a Smoothed Particle Hydrodynamics (SPH) framework, making fluid absorption and fluid flow inside the porous material possible. We show how secondary effects of the absorbed fluid on the porous material can be incorporated in existing particle-based SPH frameworks for simulating rigid bodies, elastic bodies, cloth and granular materials. This leads to the animation of new unseen effects including mass and buoyancy changes, weakening of elastic materials, sticky wet cloth, moist sand sculptures and mud formation.

To accommodate the new porous flow algorithm we build a large particle-based framework. We not only implement various existing algorithms for simulating fluid flow, elastic bodies, rigid bodies and granular materials in one unifying framework, but we also create new simulation algorithms for cloth and sand in the same unifying manner. We also present a software architecture for such a unified SPH system to provide a solid foundation for particle simulations and interactions.

In the design of these simulation models attention was spend on both the physics aspects as the creativity of the animator. The result is a physics-based animation package combining flexibility and solid simulation tools. Each of the presented simulation algorithms is therefore illustrated with several short computer animations showing the particular new effects.

Samenvatting

Computer animaties en films met speciale effecten halen tegenwoordig een zeer hoge graad aan realisme en complexiteit. Modelleer- en animatiesoftware is steeds uitgebreider geworden en biedt nu de beste algoritmes om onze echte wereld na te bootsen. Eén klasse van deze algoritmes zijn de fysisch gebaseerde methodes waarmee echte fenomenen gesimuleerd kunnen worden.

Deze thesis behandelt zulke algoritmen voor de simulatie van vloeistoffen en de dynamiek van voorwerpen. We concentreren ons specifiek op de onderlinge interacties tussen vloeistoffen en voorwerpen. Bestaande systemen simuleren typisch enkel de interactie van vloeistoffe n met de vorm van voorwerpen. Hierdoor behandelen ze het materiaal van het voorwerp als volledig ondoordringbaar. Nochtans zijn veel voorwerpen gemaakt uit poreuze materialen zoals spons of kledingweefsel, waardoor ze vloeistoffen kunnen absorberen en door hun poriën laten stromen. Dit zal daarbij hun fysisch gedrag beïnvloeden.

In dit werk stellen we een nieuw simulatie algoritme voor poreuze stroming doorheen een brede variëteit van vloeistofabsorberende voorwerpen en granulaire materialen voor. We introduceren de fysische principes van poreuze stroming, uitgedrukt door de wet van Darcy, in een SPH raamwerk. Hierdoor maken we vloeistofabsorptie en vloeistofstroming doorheen het poreus materiaal mogelijk. We tonen hoe secundaire effecten van de geabsorbeerde vloeistof op het poreus materiaal aangebracht kunnen worden in bestaande partikel-gebaseerde SPH systemen voor de simulatie van rigide en vervormbare voorwerpen, kleding en zand. Dit resulteert in de animatie van nieuwe effecten zoals veranderingen in massa en drijfvermogen, materiaalverzwakkingen, klevende natte kleding, vochtige zandsculpturen en het vormen van modder.

Om het nieuwe algoritme voor poreuze stroming te illustreren bouwen we een uitgebreid partikel gebaseerd raamwerk. We implementeren niet alleen bestaande algoritmes voor de simulatie van vloeistoffen en rigide en elastische voorwerpen in één uniform raamwerk, maar creëren ook nieuwe simulatiemodellen voor kledingstof en zand op dezelfde uniforme manier. We presenteren ook een software architectuur voor zo'n uniform SPH systeem zodat we een stevige basis voor partikel simulaties en interacties verkrijgen.

Tijdens het ontwerp van deze simulatiemodellen gaven we zowel aandacht aan de fysische aspecten als aan de creativiteit van de animator. Het resultaat is een fysisch gebaseerd animatiepakket waarin flexibiliteit gecombineerd wordt met solide simulatie tools. Daarom word ook elk van de voorgestelde simulatie algoritmes gellustreerd met meerdere korte computer animaties waarin de specifieke nieuwe effecten getoond worden.

Contents

Acronyms

2D	two-dimensional
3D	three-dimensional
ACM	Association for Computing Machinery
API	Application Programming Interface
BRDF	Bidirectional Reflectance Distribution Function
CFL	Courant-Friedrichs-Lewy
CG	computer graphics
CGF	Computer Graphics Forum
CNSPH	Corrected Normalized Smoothed Particle Hydrodynamics
CPU	central processing unit
CUDA	Compute Unified Device Architecture
DEM	Discrete Element Method
GPU	graphics processing unit
GPGPU	General-Purpose computation on the GPU
ICRI	Interdisciplinary Centre for Law and ICT
IBBT	Interdisciplinair Instituut voor Breedband Technologie
IWT	Instituut voor de aanmoediging van Innovatie door Wetenschap en Technologie
LIIR	Language Intelligence and Information Retrieval
MC	Marching Cubes
MLS	Moving Least Squares
MPS	Moving Particle Semi-Implicit

MSS mass-spring system

SIGGRAPH Special Interest Group on Graphics and Interactive Techniques

SPH Smoothed Particle Hydrodynamics

TOG Transactions On Graphics

Notation

General

a	scalar
$\mathbf{v}$	vector
$\bar{\mathbf{v}}$	normalized vector
k	constant scaling factor
$\mathbf{I}$	Identity matrix
f	Bidirectional Reectance Distribution Function (BRDF)
$\mathbf{g}$	gravitational acceleration
t	time
Δt	time step
∇	gradient
∇^2	Laplacian
$Tr(\,\cdot\,)$	trace of a matrix

Particles

p_i	particle
$\mathbf{x}_i$	position of particle p_i
$\mathbf{u}_i$	displacement of particle p_i
$\mathbf{v}_i$	velocity of particle p_i
$\mathbf{a}_i$	acceleration of particle p_i
$\mathbf{f}_i$	forces applied to particle p_i
m_i	mass of particle p_i
ρ_i	density of particle p_i
ρ_0	rest density of particles p, density in rest state
V_i	volume of particle p_i ($V_i = m_i/\rho_0$)
h_i	support range of particle p_i
$\mathbf{r}_{ij}$	distance vector from particle p_j to particle p_i, $(\mathbf{x}_i - \mathbf{x}_j)$
r_{ij}	distance between particle p_j and particle p_i, $(\mathbf{r}_{ij}/\|\mathbf{r}_{ij}\|)$
$W(h)$	normalized radially symmetric weighting kernel with support range h

Fluids

c_s	speed of sound
μ	viscosity
P	pressure
κ	surface tension coefficient

Elastic & Rigid Bodies

ϵ	strain
σ	stress
U	strain energy density
E	Young's Modulus
ν	Poisson's Ratio
$\mathbf{x}^{cm}$	center of mass
τ	torque
$\mathbf{I}$	inertia tensor
ω	angular velocity
$\mathbf{L}$	angular momentum

Granular Materials

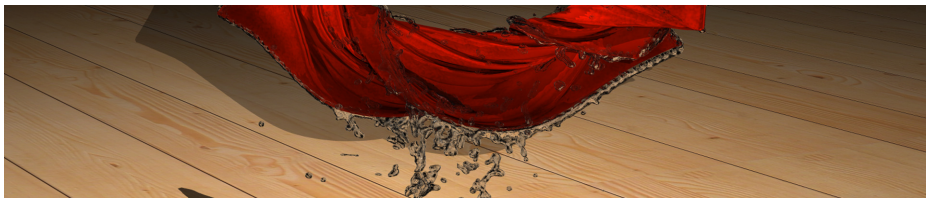
c	cohesion
μ_f	friction coefficient
ϵ	strain
σ	stress
U	strain energy density

Porous Flow

ϕ	porosity
$\mathbf{K}$	permeability tensor
K	isotropic permeability
S	saturation
P^c	capillary pressure
P^p	pore pressure
η	pore pressure scaling factor

Chapter 1

Introduction



1.1 Physics-Based Animation in Computer Graphics

Making a computer animation or a movie with special effects can take up several years. Even the shortest scenes not lasting any longer than a few seconds may need several months before modelers and animators are satisfied with the result. One of the reasons for this is that some of the visuals and behaviors of objects should be as realistic as possible (and most often directors even want something to be more spectacular than in real-life). Including real-life phenomena such as water flow, clothing, wind, smoke, explosions, fractures etc. makes this goal very hard to reach because modeling these behaviors manually quickly becomes infeasible. Therefore in computer graphics one typically resorts to physics-based animation algorithms as a starting point to simulate such effects in animations.

Physics-based animations are an exciting area of computer graphics in which physics models are applied in a virtual world on virtual objects made out of virtual materials. Often those models are simplified or approximated because the goal of the animation is not to predict the phenomenon but to trick the audience into believing the physical behavior they see is real. In addition the animator wants to retain as much control and artistic freedom as possible. For example it should be possible to make water flow into a specific direction or take a certain shape. Other reasons to simplify the physics can be speed and time to achieve interactive simulations. In other words, physics-based animations show at least *plausible* simulations of the phenomenon.

It is this area of computer graphics this dissertation is situated in. Whether one considers water splashing against the bow of boats, bath duckies dancing

on the water surface or soil grains swirling along with water flow, their interplay keeps amazing the observer. The complex dynamics of free-flow fluids are interesting to study especially when combined with other objects. Specifically those interactions in which the fluid changes the object it is interacting with form the research of this work.

At the start of our research one of the key observations in computer graphics animations was that fluids could only interact with objects at the surface, leaving the objects unrealistically dry after contact with water. At best, small-scale droplets can remain on the surface of an object [Wang et al., 2005]. Only the geometry of the objects is accounted for, treating the object as a solid. Yet not all objects are made out of impenetrable solid material; most materials are *porous* when viewed at the appropriate scale. These materials absorb and diffuse fluid through their volume upon interaction, which affects the physical properties of the material. As before, manually modeling these effects on objects is impractical. Therefore, we developed an algorithm to simulate fluid absorption and its effects on various kinds of objects including rigid bodies, deformable bodies and granular materials. Our algorithm can easily be integrated in existing fluid-object simulators. We show how absorbed water can affect the buoyancy of rigid objects, how water can weaken deformable objects, how water can make clothing wet and stick to surfaces and how water can penetrate between sand grains and form rigid sand structures or mud streams.

To accommodate such a variety of objects and materials a large physics-based animation platform is needed. This platform needs to be able to solve the Navier-Stokes equations for fluid flow, the law of Darcy for porous flow, Hooke's elasticity model and others. A unified simulation framework was developed in order to simplify interactions between these kinds of objects and fluids. In such a framework the same underlying method is used to solve the necessary equations, hence the term unified. We chose to work with SPH which is a popular method for simulating fluid flow. SPH is a particle-based integration scheme which makes a continuous evaluation of physical properties possible on points or particles discretely sampled over the volume of the fluid or object.

A unified particle framework not only allows for easier interactions including porous flow, it can be structured in such a way the animator is given more control over the simulated materials. We have designed an architecture for our framework that makes abstraction of the physical dynamics and lets animators model combinations and transitions of physical behaviors. This is needed in order to produce phase changes such as melting a wax candle, but it can also produce attractive visual effects such as characters arising from water, freezing and falling apart, or turning into sand or dust. Such effects are up till now respectively hard-coded or not possible within a single simulation system, but need a pipeline consisting of several simulation and animation packages.

In summary, this dissertation extends the interactions which can be simulated between fluids and objects in computer graphics animations. The main contri-

bution is the simulation of porous flow and its effects on various kinds of objects and volumes. Hereto, building a solid simulation framework unifying physics models using SPH was a necessity, but in the mean time provided new ways of exploring physics-based animations.

Overview

The next section lists our contributions more concretely, whereas Section 1.3 provides an overview of how these contributions are structured into the chapters of this dissertation. Important notes on the animations accompanying these chapters are written in Section 1.4.

1.2 Contributions

In the field of research on animating interactions between fluids and objects our main contributions are the following:

- **A porous flow framework to simulate fluid absorption.** We present the simulation of a fluid flowing through a porous deformable material. We introduce the physical principles governing porous flow, expressed by the Law of Darcy, into the SPH framework for simulating fluids and deformable objects. Contrary to previous SPH approaches, we simulate porous flow at a macroscopic scale, making abstraction of individual pores or cavities inside the material. Thus, the number of computational elements is kept low, while at the same time realistic simulations can be achieved. Our algorithm models the changing behavior of the wet material as well as the full two-way coupling between the fluid and the porous material. This enables various new effects, such as the simulation of sponge-like elastic bodies and water-absorbing sticky cloth.

  This work is a collaboration with Bart Adams (Katholieke Universiteit Leuven / Stanford University) and was presented as a technical paper at the annual ACM SIGGRAPH 2008 conference and published in ACM Transactions On Graphics [Lenaerts et al., 2008].

- **An SPH model for sand and fluid simulations.** We present the simulation of fine granular materials interacting with fluids. We propose a unified SPH framework for the simulation of both fluid and granular material. The granular volume is simulated as a continuous material sampled by particles. By incorporating our work on porous flow in this simulation framework we are able to fully couple fluid and sand. Fluid can now percolate between sand grains and influence the physical properties of the sand volume. Our method demonstrates various new effects such as dry soil

transforming into mud pools by rain or rigid sand structures being eroded by waves.

This algorithm builds upon [Lenaerts et al., 2008] and was presented at the annual Eurographics 2009 conference and published in Computer Graphics Forum [Lenaerts and Dutré, 2009b].

- **A unified SPH model for interactions between fluids and thin shells or cloth.** We present the two-way coupling of a fluid to thin deformable shells in a unified particle model. We use SPH for the simulation of both fluid and shells. Our cloth framework extends the work of Solenthaler et al. [2007] which uses SPH to simulate deformable and rigid volumes. Our results show realistic shell and cloth animations interacting with fluids without any leaks.

  These algorithms were developed as part of [Lenaerts et al., 2008], but were presented in parallel as an ACM SIGGRAPH poster [Lenaerts and Dutré, 2008a].

- **An architecture for Unified SPH Frameworks.** In recent years, several techniques have been proposed to simulate the physical behavior of fluids and objects. A noticeable trend in particle-based simulation frameworks is to unify the simulation of different kinds of materials, in order to simplify interactions and phase changes. SPH is often used to solve the necessary equations. While the resulting systems can simulate interactions between different kinds of materials and simulate some phase changes, they lack the freedom to model and simulate more complex animations. We propose an architecture for such a unified framework, providing not only the means to integrate multiple material simulations, but also ways to continuously fade between materials and freely combine materials. We implement our architecture as an interactive simulation and animation prototype application on the GPU. This allows the animator, within a single package, to interactively model complex animations based on physical building blocks, ranging from physically plausible phase changes such as melting wax to the fictional dynamics of paint-based clothing.

  This architecture is available as a technical report of the Katholieke Universiteit Leuven Lenaerts and Dutré [2009a].

1.3 Overview

The dissertation is structured as follows:

Chapter 2 and 3 introduce the concept of particle-based animations together with an overview of the Smoothed Particle Hydrodynamics (SPH) interpolation

scheme. We show how SPH is applied to fluid simulation.

Chapter 4 discusses how the elasticity model can be solved using SPH and applies it to different materials ranging from highly deformable to extremely stiff objects. We also extend and apply this SPH elasticity model to thin shells and cloth animations in combination with fluids.

Chapter 5 shows how granular materials such as sand can be simulated using an underlying SPH fluid. We also explore ways to render the fine grains.

Chapter 6 shows how interactions between fluids and objects and sand can be enriched by introducing porous flow to simulate fluid absorption and its effects on the object and sand volume.

Chapter 7 presents an architecture to structure the SPH models presented in Chapters 3 – 5 to form one unified SPH framework in which combinations of and transitions between materials are possible.

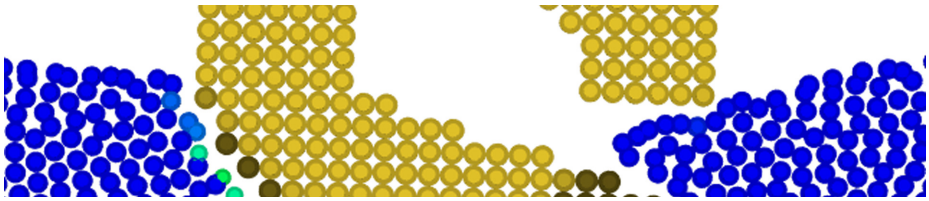
Chapter 8 formulates the conclusions of this work and provides outlooks to future research.

1.4 Notes

The figures in this dissertation contain frames from animations showing the effects of our algorithms. Though these frames were selected to be representative we would like to point the reader to the full animations. These animations are available on the enclosed DVD or can be viewed online at:
http://graphics.cs.kuleuven.be/publications/LenaertsPhD/.

Chapter 2

Particle-Based Simulation



As discussed in Chapter 1 physics-based animation is an important tool to help animate plausible visualizations of dynamic natural phenomena. This chapter elaborates on such animations and discusses the particle approach as a way to model both the object and its dynamics.

2.1 Introduction

In computer graphics to visualize a certain object or a scene containing several objects a representation of their shape is needed. One of the early and still most common representations is the *polygon mesh*. A polygon mesh is defined by a set of vertices, edges and faces by which the surface of an object is approximated. Often the faces are adjacent triangles. To animate such a mesh-based object using physics-based simulation algorithms the mesh's vertices and faces are updated conform the simulation. This works fine for objects with a clearly defined surface such as cloth or trees, but the so-called *fuzzy objects* [Reeves, 1983] present a problem. Fuzzy objects include fur, fireworks, clouds, smoke, fire and water which are difficult to model with classical mesh representations since they do not have well defined and smooth surfaces and are quite dynamic or free-form.

To simulate such phenomena the volume of the fuzzy object must be represented instead of the surface. However even objects which do have a well defined surface may need a volumetric representation in order to accurately solve the necessary physics model. Over the years many volumetric simulation methods were developed, but can typically be categorized as *Eulerian* or *Lagrangian* approaches.

In the Eulerian setting three-dimensional space is subdivided into voxels using for example a regular grid. During the simulation the algorithm then moves and/or deforms the fuzzy object through the grid. For each voxel the amount of volume of the object is computed after which the simulator decides to which neighboring voxels the volume needs to go. The basic simulation elements are thus the voxels and the object moves *through* the simulation elements.

In contrast, the Lagrangian setting subdivides the object by sampling its volume using point primitives or particles. The particles are the simulation elements which the simulator moves *along with* the dynamics of the object.

In this work we have opted for the Lagrangian approach since the goal is to extend interactions between fluids and objects. Defining multiple substances in a single grid often requires the same grid resolution for each substance. Simulating the interaction between a thin cloth and water for instance would then require a high grid resolution which would result in long computation times. Typically a different representation for the cloth such as a triangle mesh is chosen, but then interactions between different simulation elements need to be worked out. In a Lagrangian setting on the other hand interactions can be dealt with between neighboring particles, which is far more flexible than defining interfaces in a grid.

Overview

After elaborating on particle-based simulations in Section 2.2, Section 2.3 continues with a discussion of Smoothed Particle Hydrodynamics (SPH) to interpolate continuous properties from the discrete particles. Solutions for visualizing the particles volume, including surface generation and surface deformation, are provided in Section 2.4. The concluding Section 2.5 gives an overview on how SPH will be used throughout this work.

2.2 Particle-Based Simulation

The concept of particles or particle systems was first introduced by Reeves [1983] to simulate fuzzy objects in computer graphics. As mentioned earlier fuzzy objects are difficult to model with mesh representations since they have ill defined surfaces and are quite dynamic instead of limited to rigid body motion. A particle system however is essentially a point representation which provides much more freedom to model and visualize. Often no connectivity between particles is necessary (e.g., fur, fireworks, water fountains, ...), which is inherently present in mesh-based approaches. Deforming the surface mesh of a fluid which should split and merge during splashes is not trivial [Wojtan et al., 2009], but deforming a particle volume and extracting a mesh afterwards is much easier.

Particle Attributes

To model such phenomena the user can set certain global parameters controlling the overall behavior and local particle-specific parameters such as a position $\mathbf{p}$ and a color $\mathbf{c}$. A particle can have a particular mass m and a shape taking up a certain volume V which has a certain density ρ. During its lifetime t the particle may move at a certain velocity $\mathbf{v}$ through space influenced by an acceleration $\mathbf{a}$.

Particles can be created deterministically or stochastically in space before and/or during the simulation. Particle attributes can also stochastically vary to create more naturally looking effects.

Particle Dynamics

Particles are brought to life by moving them in three-dimensional space and possibly changing their attributes (for example color or volume). Using Newton's second law an acceleration $\mathbf{a}$ based on external forces $\mathbf{f}$ and the gravitational acceleration $\mathbf{g}$ can be applied to the particles:

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{\mathbf{f}}{m} + \mathbf{g} \tag{2.1}$$

Particles are then integrated in time using the forward Euler method. This boils down to moving the particles by their current velocity over a period of time dt, also called the *time step*. This time step can vary between 10^{-6}s to 10^{-3}s in our simulations (see Equation 3.15). Iterating this procedure effectively simulates the particle behavior over a larger period of time. We call one such iteration a *step*. A physics-based animation is then produced by visualizing a subset of steps in sequence. We call those particular steps *animation frames*. Typically 25 frames per second are shown in sequence creating the illusion of motion. So to produce one animation frame for a simulation time step of 10^{-3}s we need 40 iterations.

In this work we will focus on more advanced particle dynamics requiring particle-particle interactions to simulate physics models for different kinds of materials. As illustrated in Figure 2.1 we need certain particles retain the shape of a solid object, while others should flow freely over the surface of another object. To solve complex physics models a more advanced solver is needed. A popular technique for computing physical properties and forces on particles is Smoothed Particle Hydrodynamics (SPH). In the next section we provide an overview on SPH and show how it can be used to visualize and deform surfaces around particle volumes. In the next chapters we will use SPH extensively to solve the physical equations for simulating fluids, elastic materials, granular materials and porous materials.

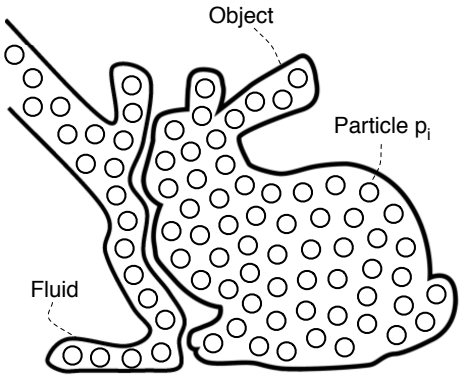


Figure 2.1: A Lagrangian simulation setup. Fluids and objects are sampled with particles on which the forces of the material model are applied. Forces are integrated in time to solve the flow of the fluids and the movements of the objects.

2.3 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics was first created to study fission in rotating stars by Lucy [1977] and Gingold and Monaghan [1977]. It has since been applied to astrophysical simulations such as star formations, solar systems, and supernovae. We refer to the work of Monaghan [1992] for a good overview.

Desbrun and Cani introduced SPH to the computer graphics community in 1996 and later Müller et al. [2003] popularized the particle method for solving fluid flows. We will discuss this application to fluids and other materials in Chapters 3 tot 5. Here we start by giving an overview on the SPH method and its principles which will be used throughout this work.

Basically SPH is an interpolation technique to provide a continuous field from the discrete particle definition. Using kernels W continuous properties $A(\mathbf{x})$ can be derived as a weighted sum of the properties $A(\mathbf{x}_j)$ defined in all particles p_j with volume $V_j = m_j/\rho_j$:

$$A(\mathbf{x}) = \sum_j V_j A(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h). \tag{2.2}$$

The function $W(\mathbf{x} - \mathbf{x}_j, h)$ is a radially symmetric kernel function with a finite support range (or smoothing length) h. W must be even (i.e. $W(\mathbf{r}, h) = W(-\mathbf{r}, h)$) and normalized over space:

$$\int_\Omega W(\mathbf{r}, h) d\mathbf{r} = 1, \tag{2.3}$$

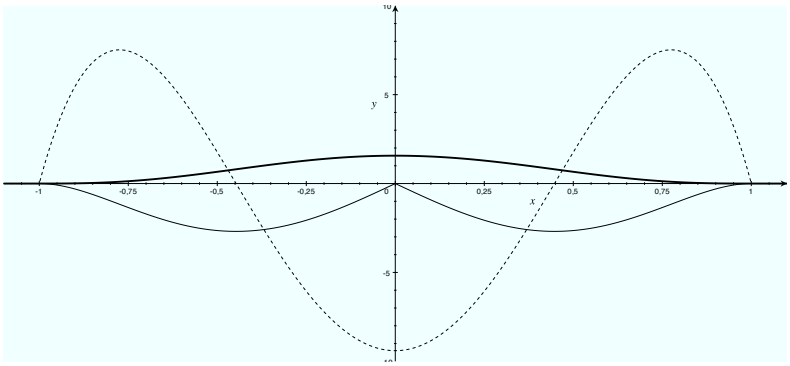


Figure 2.2: A plot of the Poly6-kernel (bold) for a smoothing length h=1, together with the first and second derivatives (respectively thin and dashed). The kernel is plotted as a radially symmetric function (i.e. $r = |\mathbf{r}|$).

where $\mathbf{r} = \mathbf{x} - \mathbf{x_j}$. Other requirements ensuring the consistency of the SPH approximations are described in [Liu and Liu, 2003]. One such smoothing kernel is the *Poly6*-kernel proposed by Müller et al. [2003]. Here, it is given in its 3D normalized form:

$$W(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r < h \\ 0 & otherwise, \end{cases} \tag{2.4}$$

where $r = |\mathbf{r}|$. A plot of the function is provided in Figure 2.2.

These radial kernels suggest spherical regions the particles occupy. Hence each particle is often represented as a sphere of radius $R_i = \sqrt[3]{\frac{3V_i}{4\pi}}$ [Desbrun and Cani, 1999].

In SPH the derivatives of a discretized function A can be obtained using the derivatives of the smoothing kernel:

$$\nabla A(\mathbf{x}) = \sum_j V_j A(\mathbf{x_j}) \nabla W(\mathbf{x} - \mathbf{x_j}, h), \tag{2.5}$$

$$\nabla^2 A(\mathbf{x}) = \sum_j V_j A(\mathbf{x_j}) \nabla^2 W(\mathbf{x} - \mathbf{x_j}, h). \tag{2.6}$$

The first and second derivatives of the Poly6-kernel are plotted in Figure 2.2.

The main advantage of using a kernel with a finite support is that Equation 2.2 and its derivatives (Equations 2.5 and 2.6) only have to be evaluated for neighboring particles (i.e. particles within the support range of the point to be evaluated) instead of all particles. Typically the smoothing length h is chosen 2 to 3 times the particle's radius. In practice range queries can be accelerated

using a grid of cells of size h. Particle neighbors can then easily be found by testing the particle's own cell and the neighboring cells. This reduces the time complexity for an SPH approximation from $O(N^2)$ to $O(NM)$, where N is the number of particles and M the average number of particles per cell.

Normalization

An SPH approximation heavily depends on the particles within the support range. When too few particles are found the value is approximated lower than the theoretical value. As an example we have approximated the function $f(x, y) = x^2 + y^2$ on a 100×100 grid of particles between $[0, 1]$ in both dimensions (Figure 2.3(a)). Figure 2.3(c) plots the error of the approximation. As can be seen in both Figures 2.3(a) and 2.3(c) the error is concentrated at the edges of the surface where less neighboring particles are found.

Normalizing the SPH approximation (Equation 2.2) leads to Corrected Normalized Smoothed Particle Hydrodynamics (CNSPH) [Vignjevic et al., 1995]:

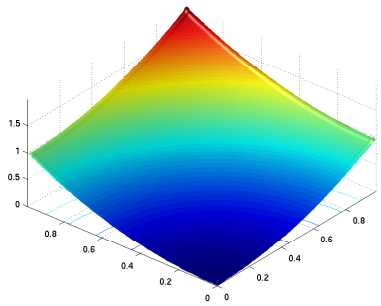
$$A(\mathbf{x}) = \frac{\sum_j V_j A(\mathbf{x_j}) W(\mathbf{x} - \mathbf{x_j}, h)}{\sum_j V_j W(\mathbf{x} - \mathbf{x}_j, h)}. \tag{2.7}$$

The resulting approximation is visualized in Figure 2.3(b) and the error of the normalized approximation is plotted in Figure 2.3(d). By normalizing the approximation a 10 times smaller error is obtained.

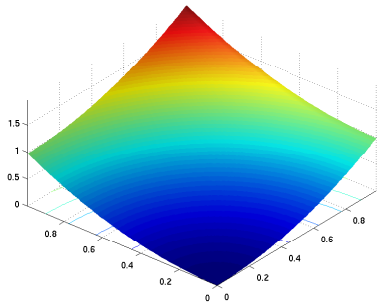
2.4 Visualization

As the next chapters will illustrate, a wide range of objects and effects can be simulated with particles. Particles sample and represent the shape of flows or objects. This shape needs to be extracted again for visualization purpose. In some cases such as for example fireworks or sparks, particles can easily be visualized by rendering colored points. However the visualization cannot always be done using particles or point primitives. Smoke for example needs a volumetric rendering and elastic objects and fluids need an actual surface. It is clear the visualization depends on the type of material that is simulated.

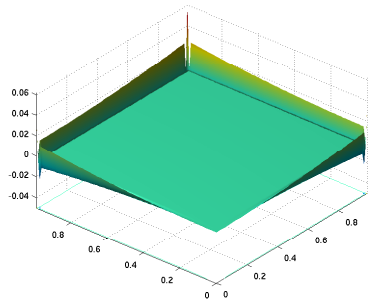
While point primitives can be used to visualize the surface [Müller et al., 2003; Adams, 2006], they require extra work like resampling the surface to avoid holes. In contrast triangle meshes are easier to render and are supported on common graphics hardware. Therefore, extracting or matching a surface mesh is far more popular and we too will focus on a visualization using surface meshes for fluids and objects.



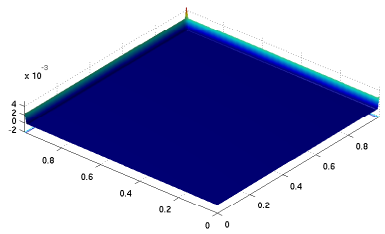
(a) SPH approximation



(b) Normalized SPH approximation



(c) Error for SPH approximation



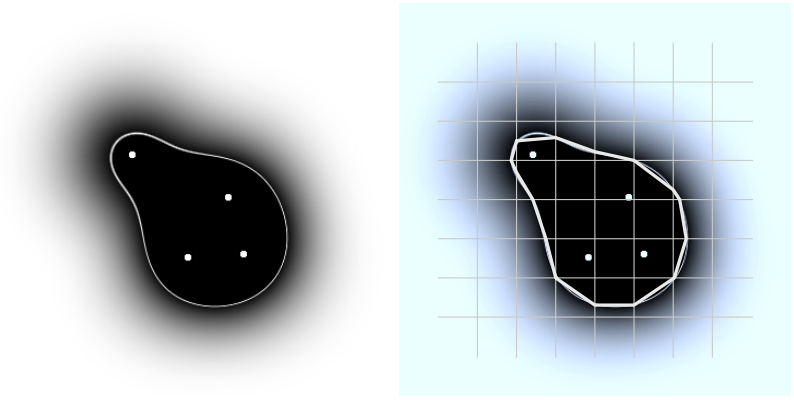
(d) Error for normalized SPH approximation

Figure 2.3: (a) An SPH approximation of the function $f(x, y) = x^2 + y^2$ and (c) the approximation error. The error is concentrated at the edges of the surface where less particles are found.(b) A normalized SPH approximation of the same function has a better accuracy (d) by weighing the approximation.

2.4.1 Iso-Surface Extraction

Typically a surface around a particle volume is extracted as an iso-contour of an implicit function defined by the particles. A *color field* is such an implicit function which is 1 at the particle locations and 0 everywhere else. The iso-level is then chosen freely between 0 and 1, depending on the desired surface (a narrow fit or rather blobby). SPH (Equation 2.2) can then be used to continuously evaluate the implicit function:

$$c(\mathbf{x}) = \sum_j V_j W(\mathbf{x} - \mathbf{x_j}, h). \tag{2.8}$$



(a) Color field and iso-surface (b) Marching Cubes mesh

Figure 2.4: (a) The color field defined by a set of particles together with an iso-contour representing the surface. (b) The surface mesh extracted using Marching Cubes (MC) on an underlying grid.

Figure 2.4(a) illustrates the iso-surface for a color field defined by a set of particles in two dimensions. One of the downsides of this approach are the so-called blobby surfaces which arise when particles are not perfectly co-planar. Solenthaler et al. [2007] build upon a method from Zhu and Bridson [2005] to try to solve this by using the positions of neighboring particles in the color field construction. Though the results from their approach are still not perfect, we use their surface definition in our work since no other approach achieves significantly better results. We refer the reader to the original work of Solenthaler et al. [2007] for the exact description of the surface definition and a comparison with the definition of Zhu and Bridson [2005].

The surface reconstruction can be done at render time in a ray tracer or as a post-simulation process using the popular Marching Cubes algorithm proposed by Lorensen and Cline [1987] to triangulate the iso-contour. In the latter case, the color field is evaluated on an underlying fixed grid and triangles are created accordingly (see Figure 2.4(b)).

2.4.2 Shape Matching

When simulating deformable objects often a detailed surface mesh is available beforehand. The mesh volume is then sampled with particles on which the simulation is performed. By linking the particle displacements $\mathbf{u}_j$ to the vertices $\mathbf{v}_i$of the mesh in a reference configuration, the mesh can be displaced as well and moves along with the simulation. Müller et al. [2004a] propose to use the SPH

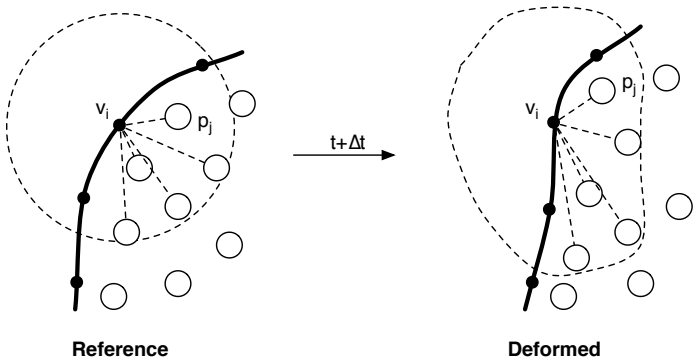


Figure 2.5: A surface mesh is displaced along with particle displacements. In the reference system vertices v_i are linked to neighboring particles p_j. During the simulation particle displacements from the reference configuration are tracked and propagated to a displacement for the vertices to create a deformed mesh.

approximation of $\nabla \mathbf{u}$ in a first order accurate approximation of the vertices' displacements:

$$\mathbf{v}_i = \sum_j (\mathbf{u}_j + \nabla \mathbf{u}_j^T (\mathbf{x}_i - \mathbf{x}_j)) \overline{W}_{ij}, \tag{2.9}$$

where

$$\overline{W}_{ij} = \frac{W(\mathbf{x}_i - \mathbf{x}_j, h)}{\sum_j W(\mathbf{x}_i - \mathbf{x}_j, h)}, \tag{2.10}$$

$$W(\mathbf{r}, h) = \begin{cases} (1 - \frac{r^2}{h^2})^3 & 0 \leq r < h \\ 0 & otherwise, \end{cases} \tag{2.11}$$

and $r = |\mathbf{r}|$. This approximation exactly matches the surface mesh to linear transformations of the particles. For large deformations including splitting of the particle volume however the surface mesh needs to be resampled, meaning triangles should be split and connected to new neighboring particles. Otherwise the triangles would stretch over the gap masking the actual behavior of the simulation underneath.

Except for fluid flows and sand volumes, we use shape matching for all objects requiring a detailed and consistent surface mesh.

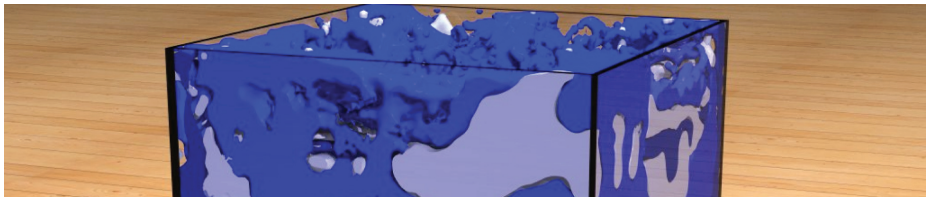
2.5 Conclusion

In this chapter we discussed SPH as an interpolation method to obtain continuous evaluations of discrete particle values. The main SPH equation (Equation 2.2) and its derivatives (Equations 2.5 and 2.6) will be used to approximate

various physics models, such as the Navier-Stokes equations for fluid flow and the law of Darcy for porous flow, throughout the following chapters. A visualization of an SPH interpolation was presented in the form of a color field. Color fields can be used to generate a surface around a particle volume to visualize in traditional mesh-based renderers. They can however serve more purposes. In the next chapters we will show how color fields can help computing the interface tension between fluids (Chapter 3), how they can advect sand grains (Chapter 5) and finally how wet regions on surfaces can be visualized using color fields (Chapter 6).

Chapter 3

Fluid Animation



Smoothed Particle Hydrodynamics (SPH) was presented in the previous chapter as an interpolation scheme for dynamic particle systems. In this chapter we will show how SPH can be used to approximate fluid behavior. More concretely, we provide an overview on how the Navier-Stokes equations which describe fluid dynamics, can be solved using SPH.

3.1 Introduction

In the last two decades, many techniques have been developed to simulate the physical behavior of fluids and objects in computer graphics. Eulerian fluid models were introduced by Foster and Metaxas [Foster and Fedkiw, 2001; Foster and Metaxas, 1997, 1996] and Stam [1999]. Over the years Eulerian fluid models were successfully coupled to rigid [Carlson et al., 2004] and deformable objects [Guendelman et al., 2005; Robinson-Mosher et al., 2008]. In recent years people have also been investigating combinations of Eulerian and Lagrangian fluid models to address the weaknesses of grid-based approaches with the strengths of particles [Losasso et al., 2008; Hong et al., 2008a; Lee et al., 2009].

Those Lagrangian methods became popular in the graphics community with the work of Desbrun and Cani [1996] and were applied to fluids by Müller et al. [2003]. The latter are typically based on the Smoothed Particle Hydrodynamics (SPH) method (see Chapter 2). Premoze et al. [2003] concurrently introduced the Moving Particle Semi-Implicit (MPS) method in computer graphics as an alternative for solving particle-based fluids. Later, Müller et al. [2005] also

started to focus on the interaction between different fluids, which was refined by Solenthaler and Pajarola [2008] to handle density contrasts at the interface. Interactions with rigid objects [Keiser et al., 2005; Becker et al., 2009b] and deformable objects [Müller et al., 2004b; Solenthaler et al., 2007] have been simulated. Clavet et al. [2005] simulated viscoelastic fluids and their interaction with rigids, whereas Krištof et al. [2009] used SPH fluids to hydraulically erode landscapes.

One of the disadvantages of particle-based fluids is they often produce compressible fluids. This issue has only recently been addressed by several authors [Colin et al., 2006; Becker and Teschner, 2007; Solenthaler and Pajarola, 2009; Sin et al., 2009].

As Müller et al. [2003] intended, SPH is perfectly suited for interactive simulations and can be seen as a research topic on its own in computer graphics. Interactive SPH simulations and visualizations of rivers on the GPU were made possible by Kipfer and Westermann [2006]. The performance of SPH fluid simulations was increased by adaptively sampling the fluid volume [Desbrun and Cani, 1999; Adams et al., 2007; Hong et al., 2008b], while others proposed SPH implementations that exploit the capabilities of current graphics hardware [Kolb and Cuntz, 2005; Hegeman et al., 2006; Harada et al., 2007]. Also combinations like adaptive resampling schemes on the GPU are investigated [Yan et al., 2009].

Overview

Fluid flow is governed by the Navier-Stokes equations, which we solve using SPH (Section 3.2). In Section 3.2.1 we discuss custom designed smoothing kernels to better approximate fluid behavior. The importance of incompressibility of fluids is shown in Section 3.3. Finally, we treat fluid simulations using multiple particle resolutions (Section 3.4).

3.2 Fluids

Central in fluid dynamics stand the Navier-Stokes equations, written down by Claude-Louis Navier and George Gabriel Stokes in the 19th century. The Navier-Stokes equations dictate the conservation of mass (Equation 3.1) and the conservation of momentum (Equation 3.2) for the dynamics of fluids. Here, the incompressible variant of the Navier-Stokes equations are given in a Lagrangian form:

$$\nabla \cdot \mathbf{v} = 0, \tag{3.1}$$

$$\rho(\frac{D\mathbf{v}}{Dt}) = -\nabla P + \mu\nabla^2\mathbf{v} + \rho\mathbf{g}, \tag{3.2}$$

where $\mathbf{v}$ is the fluid velocity, ρ the density, P the pressure, μ the dynamic viscosity constant and $\mathbf{g}$ the gravitational acceleration. Since the particles move

with the fluid, the substantial derivative $D\mathbf{v}/Dt$ is simply the time derivative of the velocity field. In a Lagrangian setting where each particle has a fixed mass Equation 3.1 can be discarded, making only Equation 3.2 of importance for particle-based fluid simulations. Analogous to Equation 2.1 this can be written as:

$$\mathbf{a} = \frac{1}{\rho}(\mathbf{f}^{pressure} + \mathbf{f}^{viscosity} + \mathbf{f}^{external}) \tag{3.3}$$

The different terms of the right hand side of Equation 3.3 are solved using SPH. Since SPH approximations require the volume of each particle (see Equation 2.2), we first need to compute the density ρ in each particle. Luckily, we can still compute this using the basic SPH Equation 2.2:

$$\rho_i = \sum_j \frac{m_j}{\rho_j}\rho_j W(\mathbf{x}_i - \mathbf{x}_j, h) = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h). \tag{3.4}$$

However, since surface particles typically have less neighbors than particles inside the fluid volume their density is approximated lower. Using the principles of Corrected Normalized Smoothed Particle Hydrodynamics (CNSPH) (see Chapter 2) we solve this by normalizing Equation 3.4:

$$\rho_i' = \frac{\sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h)}{\sum_j V_j W(\mathbf{x}_i - \mathbf{x}_j, h)}. \tag{3.5}$$

Though this requires an extra iteration over the particles ($V_j = m_j/\rho_j$), the density ρ' is used throughout the SPH solver so a better approximation benefits further SPH approximations. In the remainder of the text we will use ρ' from Equation 3.5 but simply write ρ.

The pressure P can be computed using a simplification of the ideal gas equation $P = k_p(\rho - \rho_0)$, where ρ_0 is the rest density, presented by Desbrun and Cani [1996]. This equation however can result in rather highly compressible fluids, which negatively influences the simulation (see Section 3.3). Therefore we use Tait's equation [Monaghan, 1994; Becker and Teschner, 2007] which limits the density fluctuations more strictly:

$$P = B\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right), \tag{3.6}$$

with $\gamma = 7$ and

$$B = \frac{\rho_0 c_s^2}{\gamma}, \tag{3.7}$$

where a higher speed of sound c_s enforces lower density variations. We refer the reader to [Becker and Teschner, 2007] for a description and a way to estimate the speed of sound factor to achieve only weakly compressible fluids ($< 1\%$).

The pressure force $\mathbf{f}^{pressure}$ and the viscosity force $\mathbf{f}^{viscosity}$ can be computed using the SPH derivative Equations 2.5 and 2.6:

$$\mathbf{f}_i^{pressure} = -\sum_j V_j P_j \nabla W(\mathbf{x_i} - \mathbf{x_j}, h), \tag{3.8}$$

$$\mathbf{f}_i^{viscosity} = -\mu \sum_j V_j \mathbf{v}_j \nabla^2 W(\mathbf{x_i} - \mathbf{x_j}, h). \tag{3.9}$$

However this would yield asymmetric forces, which can easily be verified for two particles. We use the symmetrized versions which do conserve linear and angular momentum as described in [Liu and Liu, 2003; Monaghan, 2005]:

$$\mathbf{f}_i^{pressure} = -\rho_i \sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W(\mathbf{x_i} - \mathbf{x_j}, h), \tag{3.10}$$

$$\mathbf{f}_i^{viscosity} = \begin{cases} -\rho_i \sum_j m_j \Pi_{ij} \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) & \mathbf{v}_{ij}^T \mathbf{x}_{ij} < 0 \\ 0 & \mathbf{v}_{ij}^T \mathbf{x}_{ij} \geq 0, \end{cases} \tag{3.11}$$

with $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and $\mathbf{v}_{ij}^T \mathbf{x}_{ij} > 0$ being equivalent to $\nabla \cdot \mathbf{v} > 0$. Π_{ij} is given as:

$$\Pi_{ij} = -\nu \left(\frac{\mathbf{v}_{ij}^T \mathbf{x}_{ij}}{|\mathbf{x}_{ij}|^2 + \epsilon h^2} \right) \tag{3.12}$$

with the viscous factor:

$$\nu = \frac{2\mu h c_s}{\rho_i + \rho_j}, \tag{3.13}$$

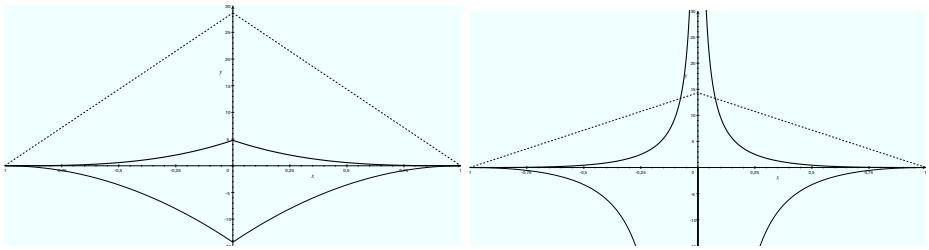
where μ is the viscosity constant. The term ϵh^2, with $\epsilon = 0.01$, prevents singularities when $|\mathbf{x}_{ij}| = 0$.

Popular and less expensive pressure and viscosity force computations were proposed by Müller et al. [2003], but in our experience Equations 3.10 and 3.11 produce more realistic results.

Surface Tension

Equation 3.6 can result in negative pressures when the approximated density is lower than the rest density ρ_0. This often occurs near the surface despite the compensation of Equation 3.5. The resulting effects on the fluid simulation are comparable with the effects of surface tension. However the speed of sound c_s is tuned towards incompressibility making the reaction to negative density fluctuations too strong. Therefore we only allow positive pressures ($P^+ = max(0, P)$) and compute the surface tension force separately according to Becker and Teschner [2007]:

$$\mathbf{f}_i^{surface} = -\kappa \frac{\rho_i}{m_i} \sum_j m_j (\mathbf{x_i} - \mathbf{x_j}) W(\mathbf{x_i} - \mathbf{x_j}, h), \tag{3.14}$$



(a) Spiky-kernel (b) Viscosity-kernel

Figure 3.1: Plots of (a) the Spiky-kernel and (b) the Viscosity-kernel (bold) together with the first and second derivatives (respectively thin and dashed). The kernels are plotted as radially symmetric functions (i.e. $r = |\mathbf{r}|$).

where κ is the surface tension coefficient. Equation 3.14 is added to the force summation in Equation 3.3.

Adaptive Time Steps

The choice of the time step in the time integration is of great importance for the stability of the simulation. We use the Courant-Friedrichs-Lewy (CFL) condition, the viscosity constant and the force terms to derive a suitable time step [Monaghan, 1992]:

$$\Delta t = min\left(0.25 \cdot min_a\left(\frac{h}{|\mathbf{f}^a|}\right), 0.4 \cdot \frac{h}{c_s \cdot (1 + 0.6\mu)}\right), \tag{3.15}$$

where $\mathbf{f}^a$ are all forces.

Intuitively this means the smaller the particles or the higher the viscosity of the fluid, the smaller the time steps and the longer the simulation takes to produce an animation frame.

3.2.1 Custom Smoothing Kernels

The choice of the smoothing kernel can influence the interpolated values, especially the derivatives of the smoothing kernel are important for the computation of the Navier-Stokes equations. Computing the pressure density force (Equation 3.10) requires the first derivate of the Poly6-kernel which drops to zero around the origin (see Figure 2.2), while high pressure forces are desired between particles moving too close to each other. Therefore, Desbrun and Cani [1996] proposed a *Spiky* kernel:

$$W^{Spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & 0 \leq r < h \\ 0 & otherwise. \end{cases} \tag{3.16}$$

As can be seen in Figure 3.1(a) the first derivative behaves as desired.

Similar for the second derivative, Müller et al. [2003] proposed a special *Viscosity* kernel to better smooth the velocity field (Figure 3.1(b)):

$$W^{Viscosity}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \leq r < h \\ 0 & otherwise. \end{cases} \tag{3.17}$$

These kernels, or rather their gradient and laplacian, are also cheap to compute. We adopt these kernels in our SPH fluid solver.

3.3 Compressible Fluids

The original SPH fluid simulation algorithms proposed by Desbrun and Cani [1996] and Müller et al. [2003] cannot guarantee incompressible fluid simulations. Using a compressible fluid solver not only results in a loss of volume, but also greatly influences the stability and the dynamics of the particle simulation. We illustrate this in Figure 3.2. A jet of fluid is aimed downwards into an empty container. We simulate the fluid using different speed of sounds c_s in Equation 3.6. The top row shows the result of using a speed of sound which allows up to 10% compressibility. The middle rows only allows 1% compressibility. The particles of the weakly compressible fluid behave much more stably and the fluid surface is much smoother. A visualization using an extracted surface mesh is given in the bottom row. The fluid is simulated using approximately 132,000 particles.

The reason why the original SPH algorithms cannot guarantee incompressible fluids, is the used pressure equation, $P = k_p(\rho - \rho_0)$, which linearly reacts to density fluctuations. Compressing a fluid will produce a pressure force which only gradually builds up the appropriate reaction, even with a high k_p constant. Becker and Teschner [2007] introduce Tait's equation (Equation 3.6) in the graphics community to faster react to these density deviations. This effectively avoids small explosions of particles in compressed regions. Particles now better maintain an even sampling pattern over the fluid volume. As a result the SPH approximations are more accurate.

For this reason we adopted the use of Tait's equation in our simulation framework. Not only our fluid simulations benefit from this adaptation, but our sand simulation algorithm (Chapter 5) and the algorithm for simulating porous flow (Chapter 6) as well. In future work, it would be interesting to look into recently proposed algorithms for incompressible particle-based fluid simulations [Solenthaler and Pajarola, 2009; Sin et al., 2009].

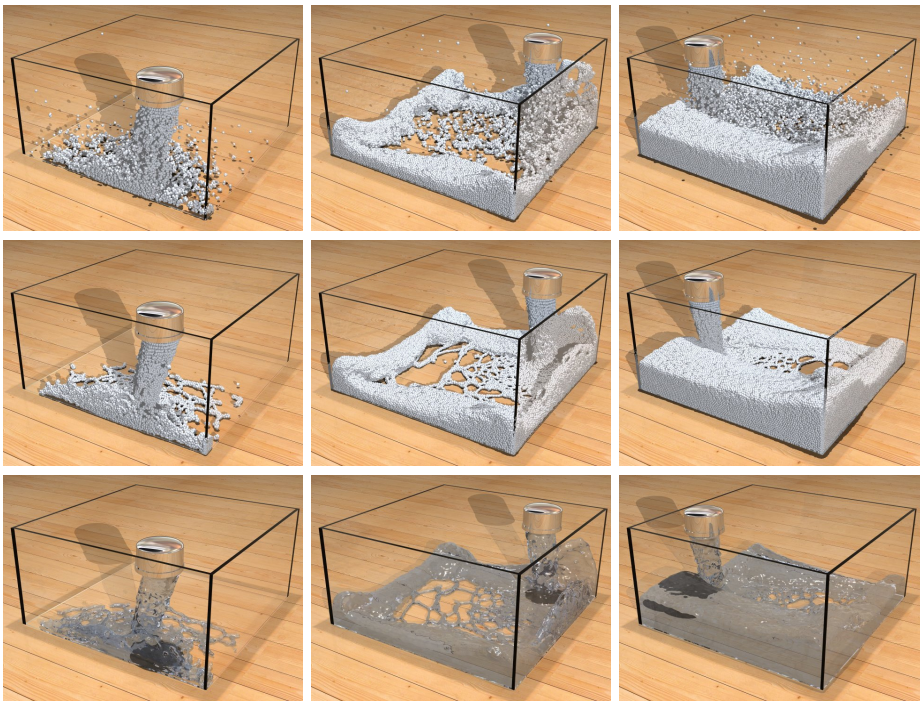


Figure 3.2: The importance of incompressibility of fluid simulations is illustrated. Compressible fluids (top row, $< 10\%$ compressible) can result in more unstable particle behavior and less smooth surfaces than weakly compressible simulations (middle row, $< 1\%$ compressible). The bottom row shows a nearly incompressible simulation visualized using an extracted surface mesh.

3.4 Multi-Resolution Particles

In a typical SPH simulation of fluid behavior all particles are of the same size. This does not necessarily have to be the case. Higher resolution particles can be used in regions requiring more detail, while in other regions lower resolution particles suffice. In this case computational effort is focussed in the desired regions, hereby effectively shortening simulation times. Being able to handle differently sized particles also comes in handy when mixing fluids with other fluids or objects. Certain objects may not need the same level of detail to adequately simulate their dynamics.

To these ends some adaptive particle models have been proposed in computer graphics. Early work by Desbrun and Cani [1999] presented an adaptive discretization of both space and time for SPH based on density differences between

particles. Adams et al. [2007] propose a similar adaptive resampling technique but use the distance to the surface as a measure.

For this work the interaction between multi-resolution particles is more important than resampling schemes and as such we will focus on the right SPH approximations in a heterogeneous particle system in this section.

Individual Smoothing Kernels

Regions sampled with higher resolution particles need smaller smoothing kernels to accurately approximate the spatial variations. Otherwise the small variations would be lost during smoothing. Therefore, the SPH smoothing length should be related to the size of the particle. Since a particle is isotropic, its volume in rest can be seen as a sphere of radius r_i:

$$\frac{4}{3}\pi r_i^3 = \frac{m_i}{\rho_0}. \tag{3.18}$$

Desbrun and Cani [1999] then propose to set the smoothing length h_i proportional to r_i:

$$h_i = \epsilon \sqrt[3]{\frac{m_i}{\rho_0}}. \tag{3.19}$$

We use a value of 1.55 for ϵ to ensure an average number of neighboring particles around particle p_i in rest conditions.

Notice that using higher resolution particles to provide more detailed simulations not only increases the total number of particles, but also requires a smaller time step (Equation 3.15) leading in both cases to longer simulation times.

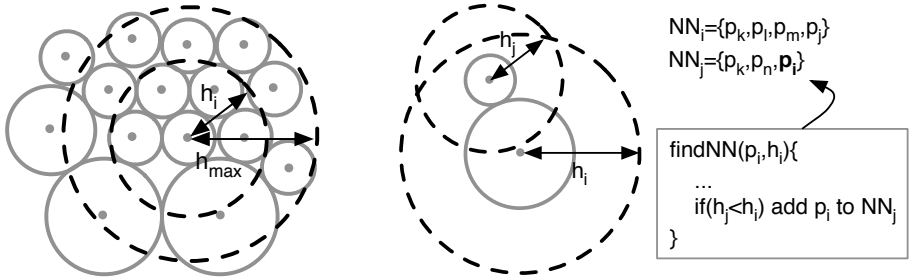
Shooting - Gathering

Using individual smoothing lengths for particles means the kernels $W(\mathbf{r}_{ij}, h)$ in Equations 3.4 to 3.14 should be replaced by either $W(\mathbf{r}_{ij}, h_i)$ or $W(\mathbf{r}_{ij}, h_j)$. The first weighting scheme is a *gathering* method since the processed particle uses its own kernel to gather the contributions of neighboring particles. The second weighing scheme is a *shooting* method since the contributions of neighboring particles are weighted according to their kernels.

Desbrun and Cani [1999] propose to combine shooting and gathering by averaging the kernels:

$$W(\mathbf{r}_{ij}, h_{ij}) = \frac{W(\mathbf{r}_{ij}, h_i) + W(\mathbf{r}_{ij}, h_j)}{2}. \tag{3.20}$$

The first and second kernel derivatives are obtained similarly. Using both shooting and gathering assures symmetric forces between pairs of particles, which shooting or gathering separately cannot guarantee.



(a) Active Query (b) Passive Query

Figure 3.3: Two types of nearest neighbor queries in a multi-resolution setting. (a) Use the largest support range h_{max} and possibly test too many particles. (b) Use the particle's support range h_i and update neighbor lists of neighboring particles when $h_j < h_i$.

It is important to note that the query for neighboring particles needs to be adapted. Two particles are now considered neighbors when $|\mathbf{x}_i - \mathbf{x}_j| < max(h_i, h_j)$. The easiest solution is to query the particles using the largest smoothing length h_{max} and then reject particles when $|\mathbf{x}_i - \mathbf{x}_j| > h_i + h_j$. In this case we are *actively* searching for particles p_j with a greater smoothing length. However, this leads to more distance tests in regions of higher resolution particles. A *passive* solution is to query neighboring particles for particle p_i using its own smoothing length h_i and add itself to the list of neighboring particles of particle p_j when p_j has a smaller smoothing length $h_j < h_i$. Both solutions are illustrated schematically in Figure 3.3.

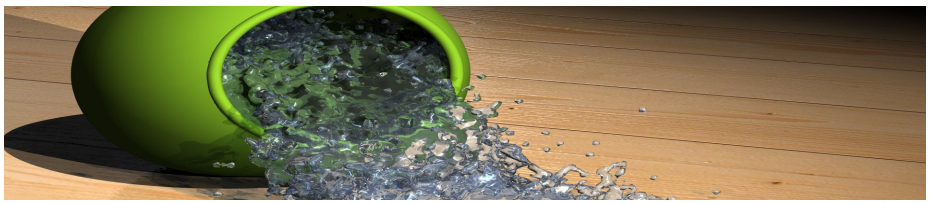
3.5 Conclusion

In this chapter we have provided an overview on fluid simulation using the simple but powerful SPH interpolation scheme. Several adjustments such as symmetrized force computations and specially designed weighting kernels increase the realism of the fluid simulation.

Our work deals with interactions between fluids and objects. It is important to realize that the sampling resolution of different kinds of objects and fluids does not necessarily have to be the same. Therefore we discussed the shooting-gathering approach to deal with multi-resolution particles. Furthermore, in Chapter 6 we will introduce porous flow into the unified simulation system which will also create multi-resolution particles during fluid absorption and emission.

Chapter 4

Elastic Animation



The freeform nature of fluids makes them an excellent medium to combine with solid objects. Objects can not only shape fluids but can also influence the fluid behavior in many interesting ways. In our daily lives we see these interactions; droplets on windows, pouring our favorite soda, a rubber ducky floating on water et cetera. Naturally in computer graphics we want to be able to simulate these interactions and effects.

It should be clear right after the simulation of fluids, the simulation of deformable objects is of great importance to this work. In this chapter we discuss an SPH approximation of the elasticity model used to simulate deformable materials. We show how this model can be extended towards thin shells and cloth simulations based on [Lenaerts and Dutré, 2008b]. Finally, we show how fluids can interact with objects in a natural way.

4.1 Introduction

One of the early models for simulating deformable objects in computer graphics was [Terzopoulos et al., 1987] which uses finite differences to solve the elasticity equations. Later they employed a mass-spring system (MSS) for simulating and melting deformable bodies [Terzopoulos et al., 1989]. Mass-spring systems are also popular for simulating cloth [Baraff and Witkin, 1998; Desbrun et al., 1999]. Tonnesen [1991] uses Lennard-Jones potential functions as another way to strengthen particles together. After their seminal work on brittle fractures O'Brien et al. introduced strain state variables to simulate elastic materials and ductile fractures [O'Brien and Hodgins, 1999; O'Brien et al., 2002].

Müller et al. [2004a] used the Moving Least Squares (MLS) procedure and particles to animate elastic objects. Later, Pauly et al. [2005] built upon their work to add fractures to deformable solids. In 2007 Solenthaler et al. ported the MLS system from [Müller et al., 2004a] to SPH and added some modifications to the reference shape of the deformable object.

Fluids interacting with objects have been studied extensively leading to several models for interactions between fluids and rigid objects [Carlson et al., 2004; Keiser et al., 2005; Clavet et al., 2005; Becker et al., 2009b], fluids and deformable objects [Müller et al., 2004b; Chentanez et al., 2006; Solenthaler et al., 2007; Robinson-Mosher et al., 2008] or even fluids interacting with granular materials [Rungjiratananon et al., 2008].

The work of Solenthaler et al. [2007] on SPH simulations of elastic materials served as a basis for all our deformable objects. In [Lenaerts and Dutré, 2008b] we proposed an extension to their work to be able to simulate thin shells in combination with fluids and presented the SPH simulation of cloth.

In the next chapters we will show how some of the principles discussed in this chapter can be used to simulate granular materials such as sand (Chapter 5) and how they are modified to be able to simulate the effects of absorbed water on the body (Chapter 6).

Overview

The elasticity model from Solenthaler et al. [2007] is discussed in Section 4.2 and serves as a starting point for the simulation of rigid bodies (Section 4.3) and our extension to be able to simulate thin shells and cloth (Section 4.4). The results are shown and discussed in Section 4.5 and a conclusion is formulated in Section 4.6.

4.2 Deformable Bodies

The method from Solenthaler et al. [2007] for modeling deformable bodies is actually an extension of the work of [Müller et al., 2004a; Keiser et al., 2005]. Instead of MLS Solenthaler et al. use SPH to compute strain and stress in the deformable body. The advantage of SPH over MLS is that it can handle coarsely sampled and coplanar particle configurations, which they needed in order to simulate melting behavior and which motivated us to formulate extensions to thin elastic shells and cloth simulations (Section 4.4).

The basic idea is to store the initial neighborhood of each particle of the deformable body in rest. This is called the *locally undeformed object condition* or the *reference neighborhood* and consists of a reference volume $\bar{v}$ and distance vectors $\mathbf{r'}_{ij}$ to the local neighbors (see Figure 4.1). During the simulation the current particle neighborhoods are compared to the reference neighborhoods.

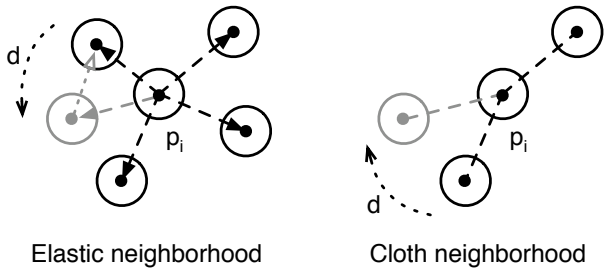


Figure 4.1: Each elastic (left) and cloth (right) particle stores a reference neighborhood (dashed vectors and lines respectively). Displacements d from this reference neighborhood translate into a strain ϵ from which the stress σ can be computed using Hooke's law. However the orientation of neighboring cloth particles is free to change, hereby allowing for folds.

We will now provide an overview on the computation of the elastic forces $\mathbf{f}^{elastic}$. For the derivation and extra details we refer to [Müller et al., 2004a; Solenthaler et al., 2007]. More information on the elasticity model can be found in [O'Brien and Hodgins, 1999; O'Brien et al., 2002].

Displacements $\mathbf{u}$ translate into an elastic strain ϵ. We calculate this 3×3 matrix using the Green-Saint-Venant strain tensor:

$$\epsilon_i = \frac{1}{2}(\nabla \mathbf{u}_i + \nabla \mathbf{u}_i^T + \nabla \mathbf{u}_i \nabla \mathbf{u}_i^T), \tag{4.1}$$

where $\nabla \mathbf{u}_i$ is the gradient of the displacement from the reference neighborhood which is computed using the SPH method:

$$\nabla \mathbf{u}_i = \sum_j \bar{v}_j \mathbf{u}_{ji}^T \nabla W(\mathbf{r'}_{ij}, h), \tag{4.2}$$

where the displacement difference vector $\mathbf{u}_{ji} = \mathbf{x}_j - \mathbf{x}_i + \mathbf{r'}_{ij}$.

From this strain ϵ the stress σ can be computed using Hooke's law, $\sigma = \mathbf{C}\epsilon$. For isotropic materials $\mathbf{C}$ only depends on Young's modulus E and Poisson's ratio υ. The stress can then be computed as:

$$\sigma_i = \frac{E}{1+\upsilon}(\epsilon'_i + \frac{\upsilon}{1-2\upsilon} Tr(\epsilon_i)\mathbf{I}), \tag{4.3}$$

where $Tr(\,\cdot\,)$ is the trace of a matrix, $\mathbf{I}$ is the identity matrix and ϵ'_i is the strain deviation:

$$\epsilon'_i = \epsilon_i - \frac{Tr(\epsilon_i)}{3}\mathbf{I}. \tag{4.4}$$

To determine the elastic force $\mathbf{f}_i^{elastic}$ of particle i, the gradient of the strain energy $U_i = 1/2(\epsilon \cdot \sigma)$ of the particle with respect to the displacement needs to be computed:

$$\mathbf{f}_{ji}^{elastic} = -\nabla_{\mathbf{u}_j} U_i = -2\bar{v}_i(\mathbf{I} + \nabla \mathbf{u}_i^T)\sigma_i \mathbf{d}_{ij}. \tag{4.5}$$

The derivative of Equation 4.2 with respect to the displacement $\mathbf{u}_j$ is also computed using SPH:

$$j \neq i \rightarrow \mathbf{d}_{ij} = \bar{v}_j \nabla W(\mathbf{r'}_{ij}, h). \tag{4.6}$$

Plasticity can easily be incorporated by testing the strain against the material's yield conditions and account for the plastic strain [O'Brien et al., 2002]. Material fractures can simply be simulated by removing distance vectors between particles in the reference neighborhoods once the material can yield no more.

4.3 Rigid Bodies

The particle simulations of highly stiff materials discussed in the previous section require small time steps to be stable. Indeed, the smallest variation from the reference shape then generates large stresses between particles which result in large elastic forces. In turn these elastic forces will try to correct the deformations, but a large time step will overshoot and make the deformation even bigger.

A common simplification for these highly stiff materials is to simulate the particular object as a *rigid body*. A rigid body discards all deformations, meaning the distance between any pair of particles in the rigid volume remains the same during the simulation.

Force acting on the particles are accumulated to a total force $\mathbf{f}^{rigid}$ and torque τ^{rigid} on the body to enforce rigid body motion, i.e. a general translation and rotation of the total particle volume [Baraff, 1997]:

$$\mathbf{f}^{rigid} = \sum_i \mathbf{f}_i, \tag{4.7}$$

$$\tau^{rigid} = \sum_i (\mathbf{x}_i - \mathbf{x}^{cm}) \times \mathbf{f}_i, \tag{4.8}$$

where $\mathbf{f}_i$ is the sum of all forces applied to particle p_i and $\mathbf{x}^{cm}$ is the center of mass of the body of particles.

The movement of the rigid body is then simulated in time by calculating the linear and angular velocity of the body. The linear velocity is obtained similar to Section 2.2. The angular velocity of a rigid body can be computed using the inertia tensor $\mathbf{I}$ and the angular momentum $\mathbf{L}$:

$$\omega = \mathbf{I}^{-1}\mathbf{L}, \tag{4.9}$$

where the angular momentum $\mathbf{L}$ is updated in every time step as

$$\mathbf{L} \leftarrow \mathbf{L} + \tau^{rigid} \Delta t. \tag{4.10}$$

In our work enforcing rigid motion is not only of importance to simulate rigid bodies, but also to update the reference shape of deformable bodies (see Section 4.2). Since the orientation of neighboring particles is not rotationally invariant, the reference shape needs to be rotated similar to the rotation of the deformable body.

This is achieved by storing the particle's initial or reference position $\mathbf{x}_i'$ and applying Equations 4.7 to 4.10 to the reference shape where $\mathbf{x}_i$ is replaced by $\mathbf{x}_i'$.

This, of course, only accounts for a total rotation and neglects rotations of subparts of the elastic body. Subdividing the elastic body in multiple parts, each having its own reference shape, is not a real solution and only complicates the matter. Recently however, Becker et al. [2009a] provided a solution to account for rotations in elastic SPH simulations.

4.4 Thin Shells and Cloth

Solenthaler et al. [2007] use SPH for its ability to handle coarsely and coplanar sampled particles and use it for melting and solidification simulations. This motivated us to apply their elastic model to thin shells and extend the model to SPH cloth simulations. Simulating thin elastic shells is straightforward by applying the elasticity model from Section 4.2 to a single layer particle sampling. However, to prevent fluids from leaking through a thin shell extra precautions are needed. We describe our measurements in Section 4.4.1 and our extension to cloth simulation in Section 4.4.2.

4.4.1 Thin Boundary

Coupling thin shells and fluids provides a challenge in avoiding leaking behavior. Only relying on fluid pressure and elastic forces as in [Solenthaler et al., 2007] is not enough to overcome leaks as high fluid pressures or low Young's moduli may lead to higher spacings between particles of the thin shell.

Therefore, we need to add an additional collision response scheme. Müller et al. [2005] simulated immiscible fluids using an interface tension force similar to the surface tension force discussed in Section 3.2. This force is always pointed away from particles of another type of object. Applying this scheme to a thin shell of particles however may not always produce a valid force vector.

We add an explicit collision handling scheme based on the boundary particle approach [Monaghan, 2005; Becker and Teschner, 2007]. A boundary force

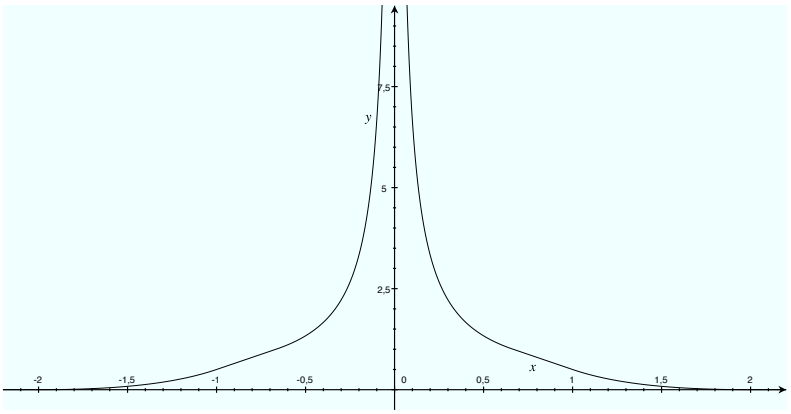


Figure 4.2: A plot of the boundary kernel Γ which increases repulsion forces when particles approach. The kernel is plotted as a radially symmetric function (i.e. $r = |\mathbf{r}|$).

$\mathbf{f}^{boundary}$ is applied between neighboring particles of which one is elastic and the other one can be anything else (fluid, elastic, ...).

$$\mathbf{f}_{ij}^{boundary} = k^b \frac{m_j}{m_i + m_j} \Gamma(\mathbf{r_{ij}}, h) \frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|}, \tag{4.11}$$

where the kernel function Γ is defined as

$$\Gamma(\mathbf{r_{ij}}, h) = \frac{1}{|\mathbf{r_{ij}}|} \begin{cases} \frac{2}{3} & 0 < q < \frac{2}{3} \\ (2q - \frac{3}{2}q^2) & \frac{2}{3} \leq q < 1 \\ \frac{1}{2}(2-q)^2 & 1 \leq q < 2 \\ 0 & otherwise \end{cases} \tag{4.12}$$

with $q = \frac{|\mathbf{r_{ij}}|}{R}$ and R the particle radius (Figure 4.2). Using the scalar k^b this boundary force can be controlled independently from the pressure force. Thus the reaction resulting from the boundary force can be made much stricter hereby avoiding leaks.

In our experience, combining such a boundary force together with an explicit collision handling scheme results in a better fluid-shell interaction. This means the influence range of a particle is divided into two regions (see Figure 4.3). When the distance between two particles is smaller than twice their radius, $r_{ij} < 2R$, their positions $\mathbf{x}_j$ are displaced along $\bar{\mathbf{r}}_{ij}$ and their velocities $\mathbf{v}_j$ are corrected:

$$\mathbf{x}'_j = \mathbf{x}_i + (R_i + R_j)\bar{\mathbf{r}}_{ij}, \tag{4.13}$$

$$\mathbf{v}'_j = \mathbf{v}_j - (\mathbf{v}_j \cdot \bar{\mathbf{r}}_{ij})\bar{\mathbf{r}}_{ij}. \tag{4.14}$$

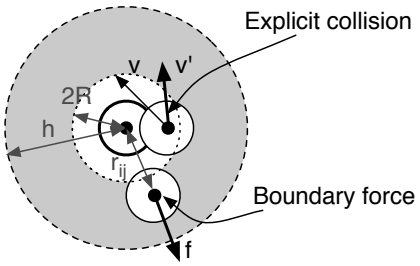


Figure 4.3: Explicit collision handling between fluid particles (thin) and shell particles (thick) is combined with a boundary force f. When the particles approach too close (within the inner white region), they are displaced and their velocities are corrected. Otherwise (gray region), the boundary force is applied.

Otherwise the boundary force $\mathbf{f}^{boundary}$ is applied between the two particles. In this case the scaling factor k^b can be chosen lower than without the explicit collision handling. Fluid particles approaching shell particles are now first slowed down. Only when there's enough pressure they will actually collide with the shell particles. We notice fluid particles are now able to slide of thin shells instead of being repelled.

Similar to [Müller et al., 2004b], boundary friction can be added using the same artificial viscosity term as for fluid-fluid interactions. Alternatively, when using explicit collision handling, the corrected velocity can be scaled down.

4.4.2 Cloth

The simulation algorithm for thin elastic shells cannot be used to simulate cloth behavior. This is illustrated in Figure 4.4. Two deformable sheets are shown, horizontally floating above a chrome sphere. The left sheet is simulated as a thin elastic shell, whereas the right sheet is simulated as a cloth. The sheets are dropped over the spheres. Notice how the elastic sheet deforms only a little and bounces on the sphere, whereas the cloth folds around the sphere. In other words, the elastic shell tries to maintain its planar shape by prohibiting bending, while the cloth allows for bends and folds. It is this kind of flexibility of cloth behavior we want to capture and add to the elastic framework.

We extend the elastic model of Solenthaler et al. [2007] to allow the simulation of cloth. As detailed in Section 4.2 they compute elastic forces for each particle using a local reference neighborhood in which each particle stores distance vectors to its neighboring particles in rest (see Figure 4.1). The extension for cloth simulation is achieved by discarding the orientation of particles in the reference neighborhood and using only the distance between neighboring particles. Doing

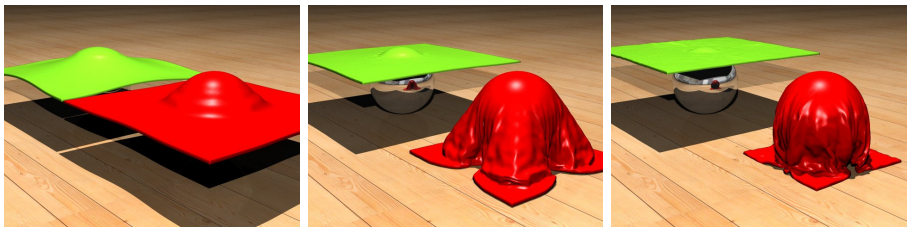


Figure 4.4: Comparison of the behavior of a thin elastic sheet and a piece of cloth. The elastic sheet tries to maintain its planar shape by prohibiting bending, while the cloth allows for bends and folds.

so appropriately allows for bending, but prohibits material stretching.

Though the elastic model tries to keep a minimum distance between particles in the reference neighborhood, self-collisions can still occur when pieces of cloth are folded and overlap. We avoid this by applying a simple penalty force proportional to the overlap between particles, hereby pushing the particles away from each other.

Particles attached to a fixed point such as the cloth in Figure 4.7 or elastic particles colliding with fluid can experience a lot of stress. Applying the artificial viscosity term (Equation 3.11) to the cloth particles basically smoothes the particle velocity field and is thus used to damp and stabilize the cloth. Because of the simplified reference neighborhood (neighbor orientations are discarded), cloth particles are less constrained than elastic shell particles and thus often need a higher damping.

4.5 Results & Discussion

Figure 4.5 shows a thin elastic bowl filled with water. The bowl is dropped on the floor which causes it to deform on impact. During the impact the water further deforms the bowl, however, without any leaks. Water only splashes through the side opening out of the bowl. The bowl consist out of 5,300 particles and contains 48,000 water particles.

A piece of cloth is dropped over 5 floating spheres in Figure 4.6. Then, water is poured on the cloth which wraps around the raising water mass until it slips through the spheres above. Also notice how the cloth can float on the water surface at the end of the animation. This is an automatic result of the pressure density forces between particles. The cloth is simulated using 15,000 particles. For the water simulation about 100,000 particles were used.

In Figure 4.7 a jet of water interacts with a piece of cloth. Notice how the force of the water jet pushes the cloth away, after which the cloth sweeps back

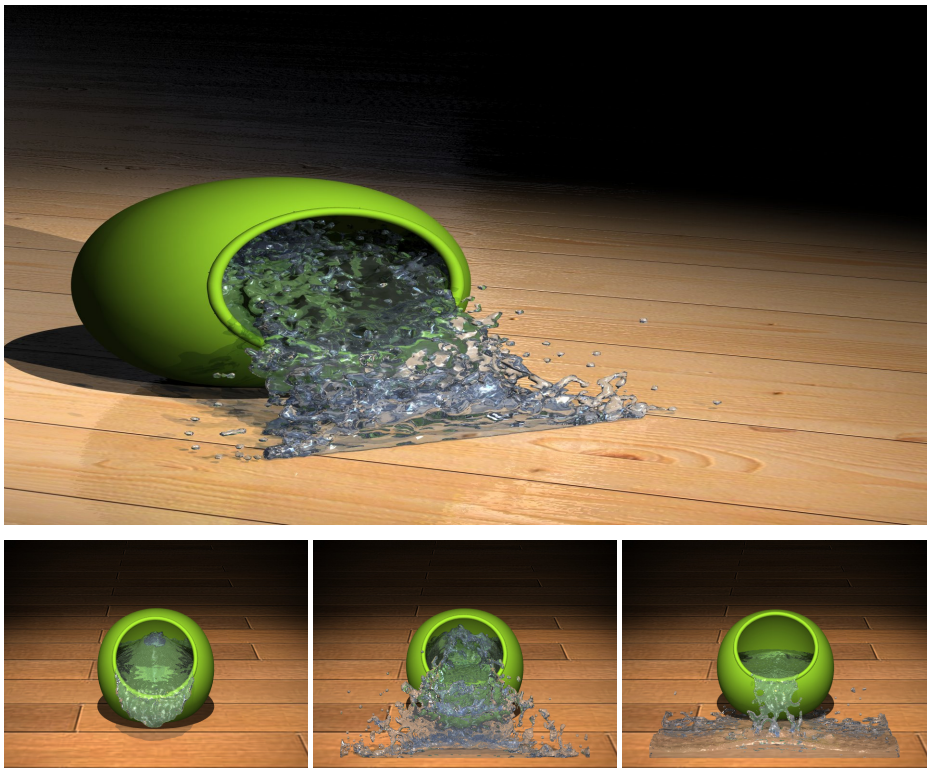


Figure 4.5: An elastic bowl filled with water is dropped to the floor. The bowl deforms on impact but water only splashes out of the bowl through the side opening.

again and influences the water flow. Also the back of the cloth is visualized to show no fluid passes through. The water is simulated using 325,000 particles while the cloth simulation only uses 15,000 particles.

Discussion

Using SPH to simulate just cloth may not be the best option. For instance the particle sampling resolution is linked to the thickness of the cloth. Mesh-based approaches are typically preferred because the simulation nodes have a fixed connectivity and the resolution can be chosen more freely. However in combination with a fluid simulation, where cloth and fluid should interact, we profit from the all particle sampling making interactions much easier than mesh based approaches.

While the boundary collision scheme provides a good coupling between fluid

Figure 4.6: A piece of cloth is dropped over floating spheres. Water is then poured on the cloth which slips through the sphere as the water mass increases.

and thin shell, there still are circumstances under which water leaks can occur. For large shell deformations fluid particles can still leak between the stretched elastic particles. We believe our method is sufficient for most animations, as the deforming bowl example (Figure 4.5) illustrates. However, in more extreme situations such as filling a balloon with water, a resampling of the elastic shell is needed to fill the gaps and solve the leaks.

One of the benefits of the SPH density based pressure model is the automatic buoyancy effect when objects interact with fluids. This can be seen in Figure 4.6 as the cloth keeps floating on top of the water at the end of the animation. In Chapter 6 we will introduce a novel porous flow model which can influence this behavior by letting the object absorb water.

4.6 Conclusion

This chapter clearly illustrates the use and benefits of SPH in simulations of deformable materials and the combination with fluids. We showed how the elastic framework of Solenthaler et al. [2007] can be extended to allow thin shell simulations in an SPH fluid environment using explicit collision handling. Furthermore, by altering their local reference shape definition, we are able to perform SPH cloth simulations. Our animations show fluid-shell interactions without leaks.

It should be clear there exists quite an amount of SPH solutions of physics models for the simulation of various types of objects and flows. In the next chapter we will add the SPH simulation of granular materials such as sand to this list and continue the trend towards a unified SPH framework for fluid, solid and granular materials.

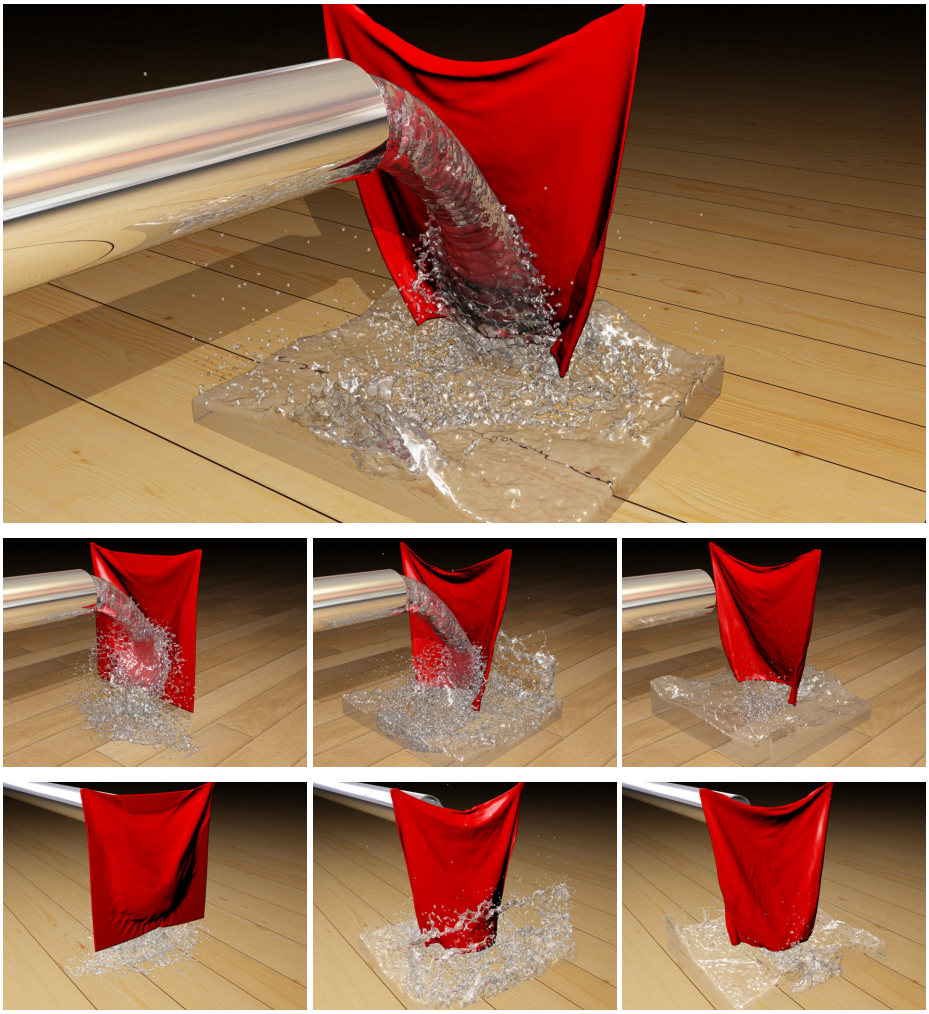
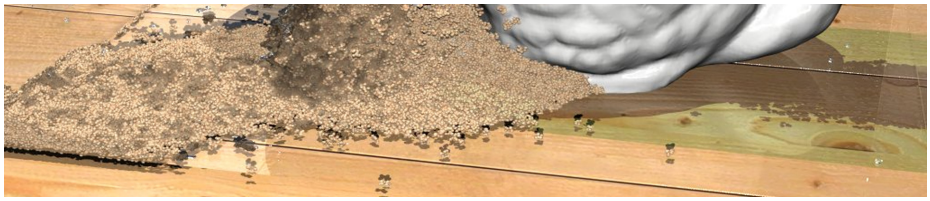


Figure 4.7: A jet of water is aimed at a piece of cloth. Water slides of the cloth which sweeps back and forth by the interplay with the jet. As can be seen in the bottom row no water passes through the cloth.

Chapter 5

Sand Animation



Based on [Lenaerts and Dutré, 2009b], this chapter expands our series of materials that can be simulated using particles and more particular using SPH. We discuss our adaptation of an existing Eulerian sand simulation method and discuss possible solutions to the challenge of rendering sand. In the next chapter we will then show how this new SPH sand simulator can incorporate interactions with fluids.

5.1 Introduction

Sand simulations in computer graphics started with dynamic terrain generation algorithms. Soil and sand terrains can be simplified and simulated as height-fields [Li and Moshell, 1993; Chanclou et al., 1996]. [Sumner et al., 1999] extended the heigh-field approach to model footprints and other tracks. Also, interactive manipulations are possible [Onoue and Nishita, 2003; Pla-Castells et al., 2008]. Sand animations based on particles were introduced by Miller and Pearce [1989]. Bell et al. [2005] simulated granular materials using a particle-particle collision model. Although they can handle large amounts of colliding bodies efficiently, the simulation resolution is directly linked to the grain size, which makes large or detailed sand simulations impractical within short time frames. In contrast, Zhu and Bridson [2005] take a continuum approach by simulating sand as a fluid, thereby decoupling the resolution of the simulation from the grain size. Wojtan et al. [2007] simulated sand erosion. Falappi and Gallati [2007] coupled granular and fluid phases using SPH. A GPU implementation of granular materials was made possible by Rungjiratananon et al. [2008]. Although they reach

interactive simulations by a GPU implementation, they simulate one sand grain by one particle and therefore are subject to the same scalability limitations as the model of Bell et al.. Alduán et al. [2009] advect high resolution particles using a low resolution simulation based on the method of Bell et al. using separated internal and external forces computations.

For a more detailed discussion of simulation models for granular materials both in computer graphics as well as in the physics community, we refer to the work of Bell et al. [2005] and Zhu and Bridson [2005].

The main contribution in this chapter is an SPH framework for granular materials. We show how sand volumes can be simulated using the sand model of Zhu and Bridson [2005] in a particle framework. As mentioned before, they employ a continuum approach which integrates much better in our simulation framework for fluids and elastic bodies than the model of Bell et al. [2005]. Looking ahead to our porous flow simulation algorithm, this method is also best suited to incorporate the secondary effects of fluid absorption. Next to simulating sand flow, we also tackle the problem of rendering sand which is both a speed and memory challenge.

Overview

In Section 5.2 we sketch the method of Zhu and Bridson and explain how this Eulerian based solver can be transfered to a Lagrangian setting. Our experiments and solution to the challenge of rendering sand are detailed in Section 5.3. Finally we discuss and conclude our work on sand in Sections 5.4 and 5.5.

5.2 Granular Materials

In our Lagrangian setting the sand volume is sampled with particles similar to a sampling of fluid volumes. Compared to earlier particle-based sand models, this continuum approach enables more efficient simulations of sand in bulk. Zhu and Bridson [2005] propose a simplified stress model to apply friction in the sand volume. A flowchart of the particle-based algorithm is provided in Figure 5.1.

First they assume the pressure inside a sand volume is similar to the pressure required to make the velocity field incompressible. This is certainly false, yet the approximation is good enough for most cases. We use our fluid SPH framework to solve for pressure gradients and make the intermediate velocity field nearly incompressible using Tait's pressure equation (Chapter 3).

Then they decompose the sand domain in regions moving rigidly and regions of shearing flow. To do this the frictional stress σ_f and the rigid stress σ_r are computed for each particle using the gradient of displacement $\nabla \mathbf{u}$ in each

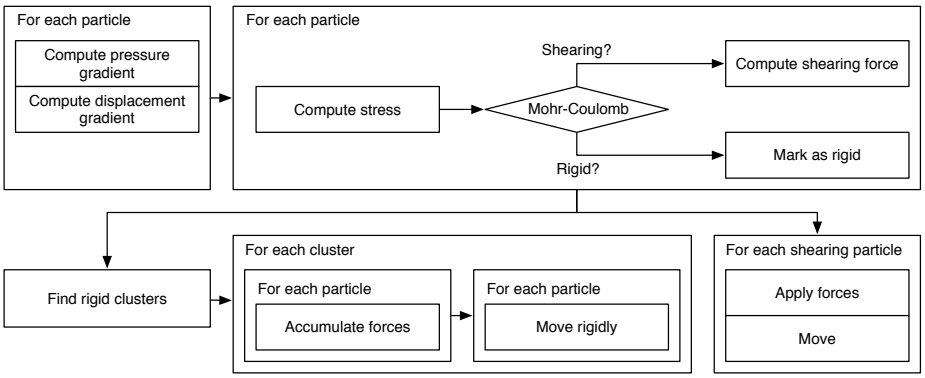


Figure 5.1: A flowchart of the particle-based sand simulation algorithm. Regions of rigid motion and regions of shearing flow are identified using a simplified stress model and updated differently.

particle. The frictional stress for particles in regions of shearing flow is given as:

$$\sigma_f = -\mu_f \frac{D}{\sqrt{1/3|D|_F}}, \tag{5.1}$$

where μ_f is the friction coefficient. The strain rate tensor $D = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$ is evaluated for each particle using SPH:

$$\nabla \mathbf{u} = \sum_j V_j \mathbf{v}_j \Delta t \cdot \nabla W(x_j - x_i, h_j). \tag{5.2}$$

Particles in regions moving rigidly can be found by testing the rigid stress σ_r:

$$\sigma_r = -\frac{\rho D h^2}{\Delta t} \tag{5.3}$$

against the Mohr-Coulomb condition, which determines material yielding

$$\sqrt{3}\sigma_f < \mu_f \sigma_r + c, \tag{5.4}$$

where c is grain cohesion.

We then search for clusters of particles marked as rigid. Two rigid particles belong to the same cluster when at least one path can be constructed between those particles over neighboring particles within a support range $h' \leq h$ (see also Figure 5.2). Smaller support ranges h' result in more rigid clusters which can break apart more easily. Forces on cluster particles are accumulated to a total force and torque and the particle cluster is then moved as a rigid body (see Section 4.3).

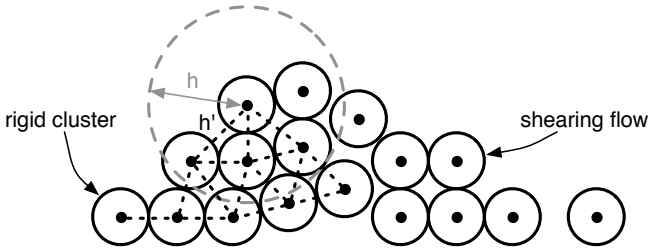


Figure 5.2: Two regions of sand flow identified by testing the Mohr-Coulomb condition: Sand particles marked as rigid are clustered and moved as a whole, the other sand particles are in a state of shearing flow and are moved separately.

The remaining particles are updated with the frictional stress by computing the force out of σ_f similar to the elastic force $\mathbf{f}^{elastic}$ (Equation 4.5).

The net result in such a sand simulation is a volume of sand in which regions stand or move rigidly and regions in which flow over the rigid regions until the total sand volume comes to rest in which case the entire volume is simulated as a rigid object.

5.3 Visualization

Once the simulation is complete the sand volume needs to be rendered. This however is a challenging problem given the size of sand grains, which diameter typically ranges between 0.05mm and 2mm. Since a major advantage of our SPH sand simulation is the abstraction of sand grains, we can't just render the simulation particles as they are typically much bigger than the actual sand grains. So we need a way to give the viewer the impression of millions of sand grains while still simulating only a few $10,000$ particles.

Extracting a surface mesh using Marching Cubes (MC) as for fluids is one solution [Zhu and Bridson, 2005], however a procedural sand texture requires texture coordinates consistent over time which is not trivial. Bargteil et al. [2006], Kwatra et al. [2007] and Mihalef et al. [2007] propose methods for texturing fluids and visualizing the flow. The later is important for sand volumes since they often show sand sliding down. In production environment sand shaders and actually simulated high resolution grains at the surface are combined, delivering more realistic results [Ammann et al., 2007], but are computationally quite expensive.

In this work we have experimented with actually visualizing the millions of sand grains on top of the low resolution simulation. We discuss our solutions in the next sections.

Figure 5.3: Advecting millions of sand grains along with the flow of a low resolution sand simulation inhibits some artifacts. Sand grains can be compressed since the sand simulation is compressible. Sand grains near the edge of the color field may suffer poor advection due to the weighting kernels.

5.3.1 Advecting Grains

An evident solution is to generate millions of sand grains in the sand volume and advect them along the velocity field of the sand simulation. This velocity field can be constructed from the particles using a color field (see Chapter 2). The color field is evaluated by placing the simulation particles on a regular grid. This can be done during the simulation or afterwards. However, while lookups in this velocity field can be accelerated using a grid structure, processing millions of grains for each time step remains a very expensive operation. Therefore, we suggest reconstructing this velocity field in a post-process.

Apart from the speed there are a few more issues with advecting sand grains. First, the velocity field is not divergence free because the simulation is based on a compressible fluid solver (see Chapter 3.2). While this is hardly noticeable in fluid simulations, visualizing the volume using minuscule grains clearly results in areas of higher density (Figure 5.3(a)).

Secondly, sand grains at the edge of the velocity field may suffer from smaller advection due to the kernel drop-off in the color field (see Figure 2.4(a)). This translates to grains 'hanging' in the air.

In very recent work, Alduán et al. [2009] address this problem. They too sample high resolution particles on a low resolution simulation, but separate internal and external forces in the eventual advection step. Only the internal forces between granular particles are being interpolated, whereas external forces such as collisions with boundaries or objects are explicitly computed for the high resolution particles. In addition, the high resolution particles are only advected with the velocity field when in the neighborhood of more than one low

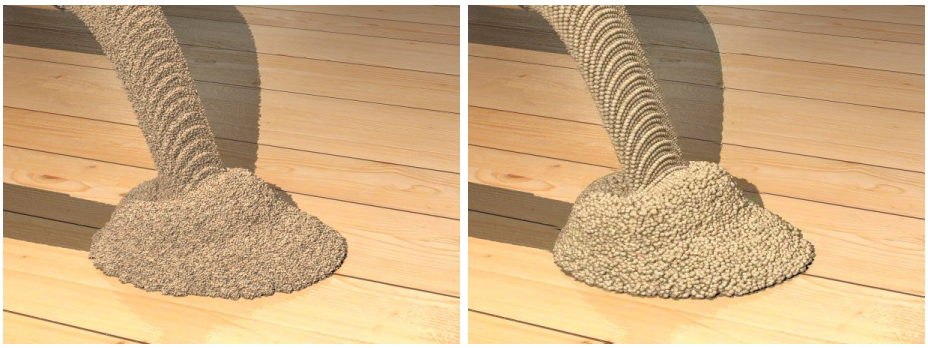


Figure 5.4: Rendering pseudo-random sand grains (left) for each simulation particle (right) conveys the illusion of sand.

resolution particle, hereby avoiding our issues at the surface of the sand volume and cluttering. Otherwise, they are fully simulated.

5.3.2 Pseudo-random Grains per Particle

To overcome the difficulties of advecting grains we employ a naive method for visualizing the simulation particles by rendering pseudo-random high resolution grains for each of the simulation particles (Figure 5.4). These high resolution grains are fixed per simulation particle to avoid flickering. This conveys the illusion of sand grains for bodies of sand and even visualizes sand flow, which would not be the case when visualizing a surface mesh.

However, the illusion fails when particles drift apart or because of a regular sampling pattern. This is noticeable in the jet in Figure 5.4. On the other hand, one of the advantages of this visualization method is that information from the simulation particles can directly be used on the sand grains. For instance, the associated sand grains of particles at the surface (which can easily be tracked) can be simulated in a post-process similar to [Ammann et al., 2007].

5.4 Discussion

Figure 5.5 shows a comparison between two simulations made with different cohesion terms c. Remember, the cohesion term in the Mohr-Coulomb condition (Equation 5.4) helps classifying particles into regions of shearing flow or regions moving rigidly. Sand is poured on the floor and piles up. The top row shows a sand simulation made with a low cohesion term, whereas the simulation in the bottom row was made with a high cohesion term. The sand pile with the

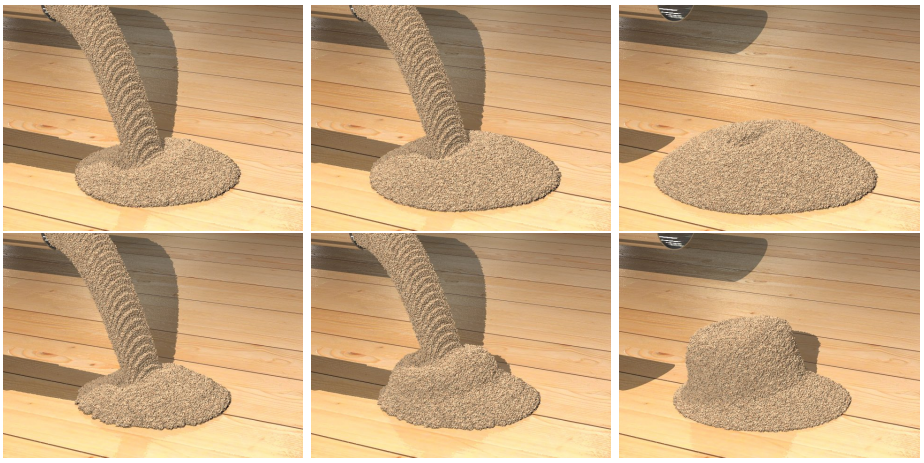


Figure 5.5: A comparison of different cohesion terms. The top row was simulated using a low cohesion term whereas the bottom row had a higher cohesion term, classifying more regions as rigid and making the sand pile rise higher.

high cohesion term is noticeably higher because more particles were classified as being rigid. This comparison illustrates the ease of use of the sand parameters, giving the animator intuitive controls to achieve the desired sand animation.

The model of Zhu and Bridson [2005] approximates the pressure in the sand by the fluid pressure used to get an incompressible velocity field for the sand volume. In some cases (e.g., hour glass simulations) this yields inaccurate results as will our method since it is based on the same principles. Our approach uses the SPH fluid method, which cannot guarantee incompressibility, and thus inherits the same compressibility issues from SPH. However, by using Tait's equation [Becker and Teschner, 2007] for the pressure computation, we obtain a weakly compressible volume ($< 1\%$), limiting the error.

In the proposed framework, one sand particle represents a volume of sand grains. Sand particles can easily drift or splash apart when interacting with fluid or objects. In that case, the sand particle is a poor approximation since the grains would probably spread. Using an adaptive particle sampling approach such as in [Desbrun and Cani, 1999] or [Adams et al., 2007] would result in a better sampling of the sand volume. Alternatively, as we discussed in Section 5.3.1 Alduán et al. [2009] address this problem too by advecting high resolution particles using a low resolution simulation using separated internal and external forces computations.

Rendering millions of sand grains appears to be a non-trivial task for existing rendering engines. We've experimented with POV-Ray [pov, 2009], Blender

[ble, 2009] and Autodesk Maya [aut, 2009] of which only the first was capable of rendering millions of parametric spheres out of the box within reasonable time. Rendering only pseudo-random sand grains at the surface and simulation particles inside the volume can decrease memory consumption and rendering times.

5.5 Conclusion

We have shown how the fluid-sand model of Zhu and Bridson [2005] can be simulated in a unified SPH framework. Simulating sand as continuous volumes enables animations consisting of larger amounts of sand compared to previous approaches.

We've also studied the visualization of sand and conclude rendering pseudo-random sand grains for each simulation particle offers a good balance between speed and conveying the illusion of sand. Furthermore it offers a nice flexibility which provides ways for post-process refinements.

Chapter 6

Porous Flow

During the previous chapters we discussed how the models for fluids (Chapter 3), rigid and deformable bodies (Chapter 4) can be simulated using SPH and freely combined to animate interactions between fluids and solid materials. In this Chapter we show how to extend this framework to handle the interaction of fluid inside of objects by simulating porous flow. We not only show how fluid can be absorbed and flow inside a porous material, but also how the absorbed fluid affects the objects.

The work presented in this chapter was the result of a collaboration with Bart Adams (Stanford University) and Philip Dutré (Katholieke Universiteit Leuven) [Lenaerts et al., 2008; Lenaerts and Dutré, 2009b].

6.1 Introduction

Although great progress has been made in fluid-solid animations (see Chapter 4), the focus has been mostly on the coupling between fluids and *impenetrable* solid objects or shells. Yet not all objects are made out of impenetrable solid material; most materials are *porous* when viewed at the appropriate scale. These materials absorb and diffuse fluid through their body upon interaction, which affects the physical properties of the deformable body. For example, a soaked spongy ball thrown at a wall reacts differently as opposed to a dry one, or wet cloths stick to surfaces due to surface tension forces of the present fluid. Granular materials such as soil or sand can also be considered as porous materials, which means fluid can percolate into the empty space between grains, affecting the physical behavior of the resulting mixture. For example, dry soil can turn into mud when

water is added, yet with the right amount of water sand castles become rigid structures that are easily destroyed by a breaking wave. These are important phenomena that can be witnessed for example on rainy days or at the beach.

To include these effects in computer graphics, not only the absorption and emission processes need to be modeled, but also the fluid diffusion within the porous material, as well as the changing material properties of the deformable objects and granular volumes.

In this work we present a novel particle method based on Smoothed Particle Hydrodynamics (SPH) to simulate fluid-absorbent deformable objects and their changing behavior. We extend our unified framework to simulate the wetting of objects; not only at the surface, but also throughout the volume of the object. Secondary effects such as the weakening and sinking of fluid absorbent objects follow immediately from the changing physical particle properties.

The main contribution of our method is the treatment of the pores at a macroscopic scale. Previous SPH methods modeling the porous body at the pore scale suffer from long simulation times and memory usage, making them not very practical and too detailed for computer graphics animations. In contrast, we reuse the particle representation of the deformable objects and add additional properties to model the diffusion process within porous materials. Collocating the elastic and porous material properties facilitates modeling of changing material behavior and results in a lightweight particle representation that can easily be incorporated in existing animation algorithms.

Overview

An overview on relevant work on porous flow is given in Section 6.2. Section 6.3 discusses how porous materials can be modeled in a particle framework. Subsequently simulating water flow inside a porous material is presented in Section 6.4. Section 6.5 then discusses water absorption and emission as a similar process as the flow simulation. The effects of absorbed water on a porous body are treated in Section 6.6. Section 6.7 deals with the visualization of wet surfaces and volumes. Finally our results are presented and analyzed in Sections 6.8 and 6.9. We conclude our work on porous flow in Section 6.10.

6.2 Related Work

The study of fluid flow through porous media is scattered throughout many fields of science. It is of importance in solid state physics, material science, geology, hydrology and petroleum engineering. A good overview of the physics of flow through porous media is given by Scheidegger [1957]. Bear [1972] was the first to present the continuum approach in modeling flow and transport phenomena. A more recent presentation on the subject was published by Hilfer [1996].

Over the years several simulation methods have been used to solve the laws governing porous flow. Lattice-Boltzmann methods [Higuera and Jiménez, 1989; Higuera et al., 1989], in which fluid particles are traced on a regular lattice, have been successfully applied to fluid flow in porous media [Ferréol and Rothman, 1995; Martys and Chen, 1996; Manwart et al., 2002]. Generally, a pore structure is modeled as solid boundary conditions on the lattice.

The use of SPH for modeling flow in porous media at a pore scale was concurrently investigated for predictive simulations by different authors. Sawley et al. [1999] presented an SPH framework in which porous media are modeled by fixed particles in the fluid domain. In a series of papers Zhu, Fox and Morris extended SPH to allow incompressible porous flow [Morris et al., 1997; Zhu et al., 1999] and describe diffusion in spatially periodic porous media [Zhu and Fox, 2001]. A Riemann-SPH method was introduced by Berry et al. [2004] to simulate contaminant transport and deposition in porous media. While these approaches result in accurate solutions, they are limited by the computational costs for practical problems as Zhu et al. [1999] concluded. The problem was alleviated by Morris et al. [1999] as they showed that SPH can be parallelized to simulate flow through porous media. However, the porous flow simulations are performed on the pore-scale and are, as such, still prohibitively expensive for use in a computer graphics context.

Besides Chu and Tai [2005] who showed how ink can be dispersed in absorbent paper using a lattice-Boltzmann approach on graphics hardware, porous flow has not been simulated before in computer graphics.

6.3 Modeling Porous Materials

A porous medium can be modeled at different scales. On the pore-scale, each of the individual pores or cavities could be carved out of the solid phase. In this work, we opt for an efficient representation at a macroscopic scale. We remodel rigid or elastic particles as *porous particles* and let them represent a small porous volume, capable of holding an amount of fluid (Figure 6.1). This unified representation allows us to use the same particle resolution as for regular solid-fluid simulations. Moreover, it allows modeling changing material behavior when fluid is absorbed by a particle.

A porous particle is completely defined by its *porosity* and *permeability*. The porosity ϕ_i denotes the volume fraction of the interconnected void space at the particle, i.e., $\phi_i V_i$ denotes the void particle volume. Hence, a particle p_i with porosity ϕ_i can hold an absorbed fluid mass $m_{pi} \leq \rho^{fluid}\phi_i V_i$ which defines the particle's saturation S_i:

$$S_i = \frac{m_{pi}}{\rho^{fluid}\phi_i V_i}. \tag{6.1}$$

Note that m_{pi} denotes the fluid mass in the porous particle, while m_i denotes

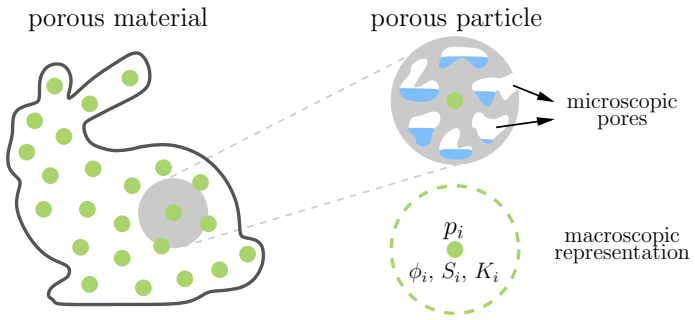


Figure 6.1: Porous materials are represented at a macroscopic scale. Pores or cavities of a certain region in the volume are not modeled explicitly. Instead, the region is represented at a coarser level by the use of porous particles (right), which have porosity and permeability parameters that characterize the surrounding material.

the particle's own unsaturated mass.

The *permeability* $\mathbf{K}$ of a porous medium characterizes its ability to transmit a certain fluid. $\mathbf{K}$ is a second-order symmetric tensor, which may have non-zero off-diagonal elements if the porous medium is anisotropic. It reflects the actual internal distribution of pores for a certain region in the material. In our work, we focus on isotropic porous materials, hereby reducing the permeability tensor $\mathbf{K} = K\mathbf{I}$ to a scalar. We assign a permeability K_i to each porous particle p_i and are therefore able to model both homogeneous and heterogeneous porous media.

6.4 Simulating Porous Flow

For small-scale fluid dynamics, surface tension (or interfacial tension) is of great importance. At the pore-scale, the entrance of fluid into a pore is facilitated by the surface tension because of a pressure differential between the entering fluid phase and the displaced phase inside the porous medium, which can be air or another fluid. This pressure, called the capillary pressure, makes fluid diffuse to neighboring, less saturated pores. These forces also keep the fluid trapped inside the medium. Since we have no microstructure of the pores, we model the capillary forces acting on the fluid represented in a porous particle as gradients of capillary potentials [Nitao and Bear, 1996; Hilfer, 2006], which we compute using SPH:

$$\nabla P_i^c = \sum_j V_j P_j^c \nabla W(\mathbf{x}_j - \mathbf{x}_i, h_j), \tag{6.2}$$

where the capillary potential P^c is defined as a pressure function of the saturation S:

$$P_i^c = k^c(1 - S_i)^\alpha, \tag{6.3}$$

with constants k^c and $0 < \alpha < 1$ controlling the strength of the potential. The capillary force of the pore space of a porous particle drops as its saturation increases.

Another important pressure difference can arise when deformations of a porous elastic body change the volume of the pore space. These porosity changes may exert a pressure on the present fluid, causing it to flow out of the squeezed pores. This leads to a variable porosity ϕ_i depending on the local density of the medium:

$$\phi_i = \phi_0 \frac{\rho_0^s}{\rho_i^s}, \tag{6.4}$$

where ϕ_0 is the porosity in rest. We compute the pore pressure P_i^p similar to the equation of state as in Becker and Teschner [2007]:

$$P_i^p = k^p S_i \Big(\Big(\frac{\rho_i^s}{\rho_0^s} \Big)^\gamma - 1 \Big). \tag{6.5}$$

The saturation level S_i is added as a factor, motivated by our macroscopic approach: Fluid may flow inside the volume of one porous particle as long as it is not fully saturated.

These pressure differences give rise to the Darcy flux [Darcy, 1856], which governs the flow of an incompressible, single-phase fluid through a porous medium. It can be derived from the more general Navier-Stokes equations and relates a pressure gradient ∇P on a fluid with viscosity μ and completely filling a porous medium with permeability K to a flux. This Darcy flux is related to the pore velocity $\mathbf{v_p}$ by dividing by the porosity ϕ:

$$\mathbf{v_p}_i = -\frac{K_i}{\phi_i \mu} (\nabla P_i^p - \nabla P_i^c - \rho \mathbf{g}). \tag{6.6}$$

This pore velocity field defines an anisotropic diffusion inside the medium. In contrast to the fluid particles outside of the porous medium, we model the fluid flow inside the medium in a Eulerian way using a diffusion process. Fluid mass is diffusing from one porous particle to the next, rather than tracing fluid particles through the medium. The SPH approximation to the diffusion equation [Müller et al., 2005] for the evolution of the absorbed fluid mass m_p is:

$$\frac{dm_{pi}}{dt} = \sum_j d_{ij} V_j m_{pj} \nabla^2 W(\mathbf{x}_j - \mathbf{x}_i, h_j), \tag{6.7}$$

in which we define the diffusion coefficients d_{ij} between two particles proportional to the direction and length of the pore velocity:

$$d_{ij} = \mathbf{v_p}_j \cdot \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} S_j^\beta. \tag{6.8}$$

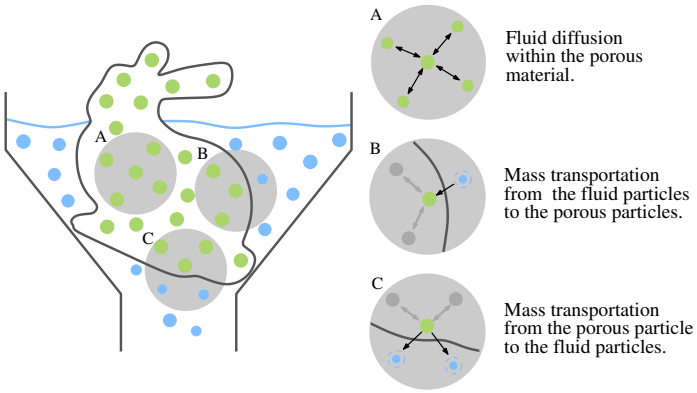


Figure 6.2: Three different cases for fluid transportation in the presence of porous materials. A: Within the porous object fluid mass is diffused at a macroscale between the porous particles. B: Fluid particles near the porous material are treated as porous particles and mass is taken away from them if fluid is absorbed by the porous material. C: When the porous material emits fluid, mass is added to neighboring fluid particles, which are again treated as porous particles. If, after the diffusion process, more fluid remains to be emitted, new fluid particles will be created in the neighborhood of the porous particle.

Similar to [Bear, 1972; Cuesta et al., 1999; Hilfer, 2006], we make the diffusion dependent on the saturation S_i of a porous particle. The user defined $\beta > 0$ is a parameter to control this flow. It can be interpreted as internal flow, where pores inside the particle volume are filled before flowing to neighboring particles.

Finally, the fluid mass is integrated in time using an explicit Euler integration step:

$$m_{pi} \leftarrow m_{pi} + \Delta t \frac{dm_{pi}}{dt}. \tag{6.9}$$

Due to the Eulerian nature of the diffusion process, mass conservation is not automatically guaranteed and the total diffused fluid mass must be explicitly controlled. We do this by explicitly checking the diffusing fluid mass against the free and occupied volumes, $(1 - S_i)\phi_i V_i$ and $S_i \phi_i V_i$ respectively, inside the porous particles.

6.5 Porous Medium-Fluid Coupling

At the contact surface between a porous material and the surrounding fluid, porous particles interact with fluid particles. The surrounding fluid is absorbed using particle deletion and absorbed fluid can be emitted back into the surrounding fluid by dynamic particle creation. We now discuss these absorption

and emission processes.

6.5.1 Absorption

Fluid particles near the surface of a porous body may be absorbed due to capillary forces. We use the same diffusion process as described in the previous section by treating the fluid particles at the surface as saturated porous particles with a porosity $\phi_i = 100\%$ and saturation $S_i = 1$ in Equations 6.2 to 6.9. These fluid particles are included in the diffusion process and are deleted as soon as all their mass is diffused in the absorbing material (Figure 6.2 B).

The gradual absorption of the fluid particle mass avoids discontinuities that would arise if we would absorb the fluid particles at once and decouples the resolution of the fluid from the resolution of the porous body. However, it introduces variable sized fluid particles in the fluid simulation framework. As discussed in Chapter 3, we employ the shooting-gathering approach of Desbrun and Cani [1999] to obtain symmetric forces between particles with different smoothing lengths. A variable time step based on the Courant condition for the smallest h_i stabilizes the simulation. A threshold for the particle volume, typically down to 1% of the original size, is enforced to guarantee a minimum time step.

6.5.2 Emission

When the pore velocity field $\mathbf{v_p}$ pushes the fluid outside the medium, fluid mass has to be transferred from the porous material to the surrounding fluid (Figure 6.2 C). Similarly to the absorption mechanism, we treat fluid particles at the interface as unsaturated porous particles ($S_i = 0$) and let absorbed fluid mass diffuse to the fluid particles as described in Section 6.4. These fluid particles may grow in volume until they reach the normal size of a fluid particle.

If no fluid particles surround the porous material or if additional fluid mass remains to be emitted, new fluid particles are created as follows: The pore velocity field $\mathbf{v_p}$ defines the positions for these new fluid particles for the current time step. To avoid large pressure forces between neighboring fluid particles, we follow the approach of Adams et al. [2007] to maximize the distance to existing particles. If there is a particle too close (within a threshold distance), we look for a new position by random sampling around this target position within a sphere with half the particle radius. Ensuring this minimal particle distance results in a more stable particle creation and emission process. Moreover, new fluid particles are usually relatively small at creation and will push neighboring particles aside as they expand due to diffusion in the next time steps.

These absorption and emission processes are illustrated by the 2D simulation in Figure 6.3 where the simulation particles are visualized. A porous S-shape is dropped into a tank of water. Water particles are color-coded depending on

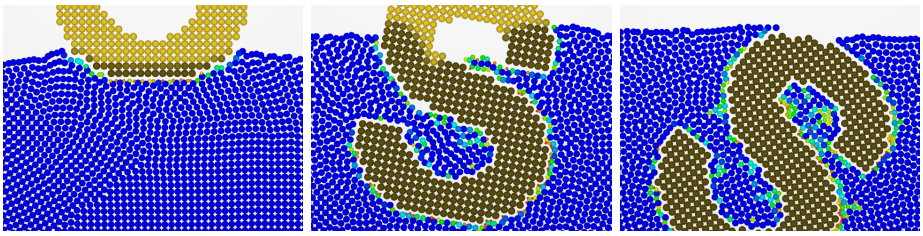


Figure 6.3: A particle visualization of the absorption and emission processes during the simulation of a porous S-shape dropped into a tank of water. Water particles are color-coded depending on their mass; the original blue particles turn to red as they loose mass during the absorption process. Other smaller particles turn back to blue as they grow in size due to the emission process.

their mass; the original blue particles turn to red and shrink in size as they loose mass during the absorption process. Notice how some small particles turn back to blue as they grow in size due to the emission process. Since plenty smaller particles are available at the interface no new particles need to be emitted in the emission process, keeping a steady particle count.

6.6 Modeling Changing Material Properties

Fluid absorbed by a porous object affects its material parameters and hence its behavior. Particles gain weight, implying the solid mass m_i and absorbed fluid mass m_{pi} of a porous particle should be used in all computations. The density ρ_i of a porous particle is called the bulk density. The bulk rest density becomes:

$$\rho_{0i} = \rho_0 + S_i \phi_i \rho_0^{fluid}, \tag{6.10}$$

to account for the saturated pores in the equation of state. The net result for a soaked body is a higher mass and density and thus changed physical behavior as compared to the dry material.

6.6.1 Elastic Bodies

When simulating elastic bodies, the fluid pressure also reduces the stress σ_i in the body:

$$\sigma_i^{eff} = \sigma_i - \eta P_i^{p*} \mathbf{I}. \tag{6.11}$$

Equation 6.11 is called the *effective stress* and determines material weakening and volume gain. The $P_i^{p*} = k^p S_i$ represents the pore pressure of the present fluid. Resulting from our experience we added a scaling factor η to the equation

to control the balance between porous flow (Equation 6.5) and the influence of the pore pressure on the elastic body.

6.6.2 Granular Materials

A granular material can be considered as a porous material. The space between individual grains of a granular volume is the actual pore space. Work has been published on simulating granular materials and fluids separately (see Chapter 5), but fully coupling the two volumes seems difficult. One challenge is that fluid has to be able to flow through the open space in the granular volume, which directly couples the fluid and sand resolution at which the simulation is performed. Another difficulty is simulating of the resulting mixture, since the fluid can alter the behavior of the granular material or vica versa.

By choosing a simulation algorithm that simulates on volumes of grains instead of simulating the actual grains, we can use porous particles as volumes of grains. Moreover, simulating sand as a fluid allows us to easily simulate a mixture of granular material and fluid and transition from dry dirt to a mud stream for example.

In a moist sand volume the surface tension of the fluid between the sand grains strengthens the sand volume to form a more rigid structure. At this point the sand volume has reached an ideal saturation level S'. We use the saturation level to control the cohesion term c in the Mohr-Coulomb condition (Equation 5.4). We linearly interpolate between a cohesion c^{dry} for the dry material and a cohesion c^{wet} for the moist material, depending on the saturation level (Figure 6.4):

$$c = \begin{cases} c^{dry}(1 - \frac{S}{S'}) + c^{wet}\frac{S}{S'} & \text{if } S \leq S' \\ c^{wet}\frac{1-S}{1-S'} & \text{if } S > S' \end{cases} \tag{6.12}$$

As more water is added to the mixture, sand grain spacing increases and the mixture becomes more liquid. To simulate such a mixture we add a viscosity term [Müller et al., 2003] to the velocity field of the sand and use the saturation level S to scale the viscosity coefficient $\mu' = \mu(S - S')/(1 - S')$ of the fluid-sand mixture. At the same time, we down-scale the friction coefficient $\mu'_f = \mu_f(1 - S)/(1 - S')$ of the sand as the saturation increases.
The net result is a sand volume that can turn in to a rigid structure or a viscous fluid volume and vice versa. The evolution of these parameters versus the saturation is plotted in Figure 6.4.

6.7 Visualization

The appearance of objects changes when water or another fluid is absorbed. In general, wet materials look darker compared to dry materials due to increased

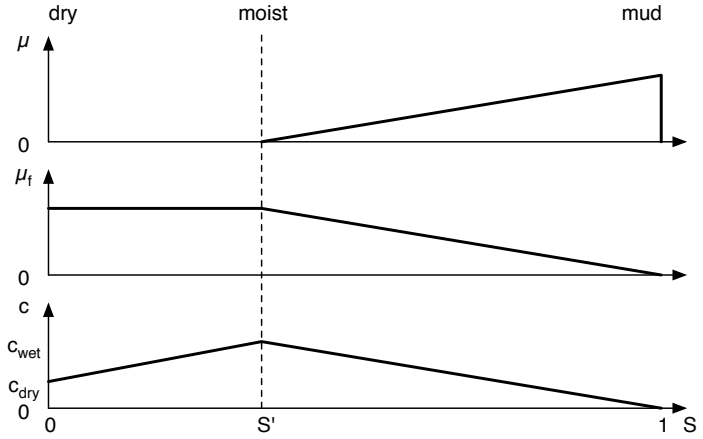


Figure 6.4: As fluid is absorbed, the saturation level S determines sand cohesion c, sand friction μ'_f and viscosity μ'. Dry sand transitions to a rigid moist structure to viscous mud.

light absorption [Twomey et al., 1986]. Although complex models for the rendering of wet surfaces with full subsurface scattering are available [Jensen et al., 1999], we use a simple and intuitive model for visualizing our results. We model the absorption using a darkening factor δ $(0 < \delta < 1)$ for wet regions. If f_r^{dry} is the Bidirectional Reflectance Distribution Function (BRDF) of the completely dry material, the BRDF of the wet material, f_r^{wet}, is approximated at particle p_i by

$$f_{ri}^{wet} = S_i\phi_i(\delta f_r^{dry}) + (1 - S_i\phi_i)f_r^{dry}, \tag{6.13}$$

where δf_r^{dry} is the completely wet material and $S_i\phi_i$ is a measure for the wetness. A continuous BRDF evaluation at the vertex positions $\mathbf{x}$ is obtained using Equation 2.2:

$$f_r^{wet}(\mathbf{x}) = \sum_j V_j f_{rj}^{wet}(\mathbf{x}_j)W(\mathbf{x}_j - \mathbf{x}, h_j). \tag{6.14}$$

As detailed in Section 5.3 we visualize sand volumes by rendering pseudo-random high resolution particles for each of the simulation particles. In the case of wet sand and soil the appearance of the surface becomes much smoother. Especially when transitioning to mud it is important to visualize a liquid-like surface. Therefore, as the sand volume saturates we gradually fade in a sand mesh to illustrate the wet look. This is achieved by using the saturation level in the alpha component of the vertex colors. High resolution sand grains are darkened by applying Equation 6.13 to the corresponding simulation particles, the surface mesh is darkened using Equation 6.14 as before. Figure 6.5 shows

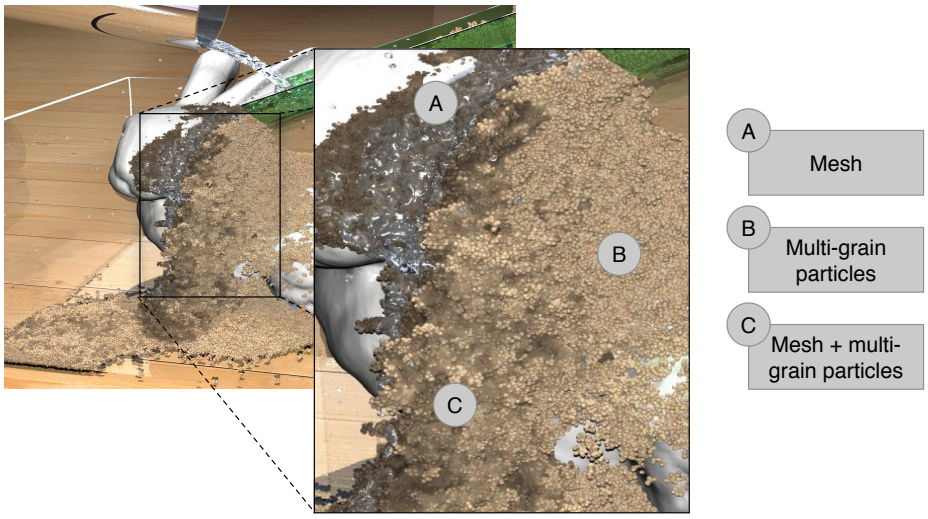


Figure 6.5: A close-up of a rendering of wet sand. Three regions can be identified; (A) a dark surface mesh visualizes the wet sand, (B) multi-grain particles are used to render the dry sand, (C) a blend of the surface mesh and multi-grain particles illustrates the moist sand.

the result of such a rendering featuring dry, moist and wet sand. Sand surfaces are extracted using the colorfield of Solenthaler et al. [2007] and the Marching Cubes algorithm [Lorensen and Cline, 1987], as described in Section 2.4.1.

6.8 Results

In all animations we set the capillary strength k_c to $15,000$Pa and α and β respectively to 0.1 and 7. We use a value of $0.0\text{Pa}\cdot\text{s}$ for the artificial viscosity of water Monaghan [2005]; Becker and Teschner [2007]. Full animations can be seen in the accompanying video. Not all simulations were computed on the same hardware, we provide an overview of the timings and used hardware at the end of the section.

Figure 6.6 shows a comparison of the porous flow parameters by pouring water against a dry sponge, measuring 5cm×5cm×5cm. The green sponge has a 70% porosity and a permeability of $1e^{-12}\text{m}^2$. Notice how the percolated fluid affects the volume and stiffness of the sponge. The pore pressure constant k^p is set to $5e^4$Pa for the green sponge, whereas the red sponge has a k^p value of $7.5e^4$Pa and thus has a larger volume gain. The orange sponge's permeability is 10 times lower than the one of the green sponge causing the fluid to flow at a slower pace.

The twice as high capillary pressure of the blue sponge causes the fluid to rise up higher in the sponge volume. The sponges are simulated using 5,000 porous particles, the water is simulated using 25,000 particles.

Figure 6.7 shows a tank filled with a heterogeneous porous material with varying permeabilities. The porous medium's dimensions measure 10cm ×10cm ×10cm. An S-shaped region in the porous medium has a permeability of $1^{-12}\mathrm{m}^2$, whereas the other regions have a permeability of only $2e^{-15}\mathrm{m}^2$. The porosity does not vary and amounts to 50%. As expected, the regions of higher permeability saturate first as the fluid flows down the tank. Capillary forces pull the fluid to the right in the lower part of the S-shape. 32,000 particles are used to simulate the porous medium and the fluid on top is simulated by maximum 32,000 particles. The average computation time per frame is 2 minutes.

Figure 6.8 shows an animation of a wet cloth's tendency to stick to surfaces due to high surface tension forces caused by a small layer of fluid between the surface and the cloth. We simulate this behavior by increasing the adhesion forces between cloth particles and surfaces and make the friction proportional to the saturation level of the porous particle. The result can be seen in the top row of Figure 6.8 in which a 30cm×40cm wet velvet cloth is draped around a sphere and starts dripping. The cloth is sampled by a single layer of 20,000 particles that have a uniform porosity of 70%, a permeability of $1e^{-12}\mathrm{m}^2$ and are 80% saturated in the beginning of the animation. Notice how the cloth folds and sticks around the sphere due to the present fluid, in contrast to a dry cloth which can be seen in the accompanying video. The bottom row of Figure 6.8 shows frames from an animation where the same cloth is twisted to wring the fluid out of the cloth. As the cloth folds and tightens, the porosity is reduced and the water naturally flows out of the cloth. Approximately 80% of the fluid in the porous particles is emitted at the cloth surface, hereby creating 25,000 fluid particles. The computation time per frame was on average 1.5 minutes for both animations.

Figure 6.9 shows an animation in which water inside a porous Armadillo model is squeezed out. The Armadillo is sampled with 30,000 porous particles of 70% porosity and saturated for 95%. Capillary forces hold the fluid mass in the porous volume unless the porous particles are deformed (e.g., around the fingers). As the hand closes the volume of the Armadillo is reduced by 15% on average and the resulting pore pressure pushes the fluid out. Notice how emitted fluid can be absorbed again at a different point. While the hand squeezes the number of particles increases to 20,000 and drops again to 10,000 when the hand stops moving. The computation time was approximately 20 minutes per frame due to a small simulation time step that was required to ensure adequate squeezing behavior avoiding hand-Armadillo penetrations.

As presented in Section 6.6.1 the fluid pressure in the pores absorbs some of the stress of the pore-elastic body. The resulting behavior of the effective stress is illustrated in Figure 6.10. A dry Stanford Bunny consisting of 24,000 porous

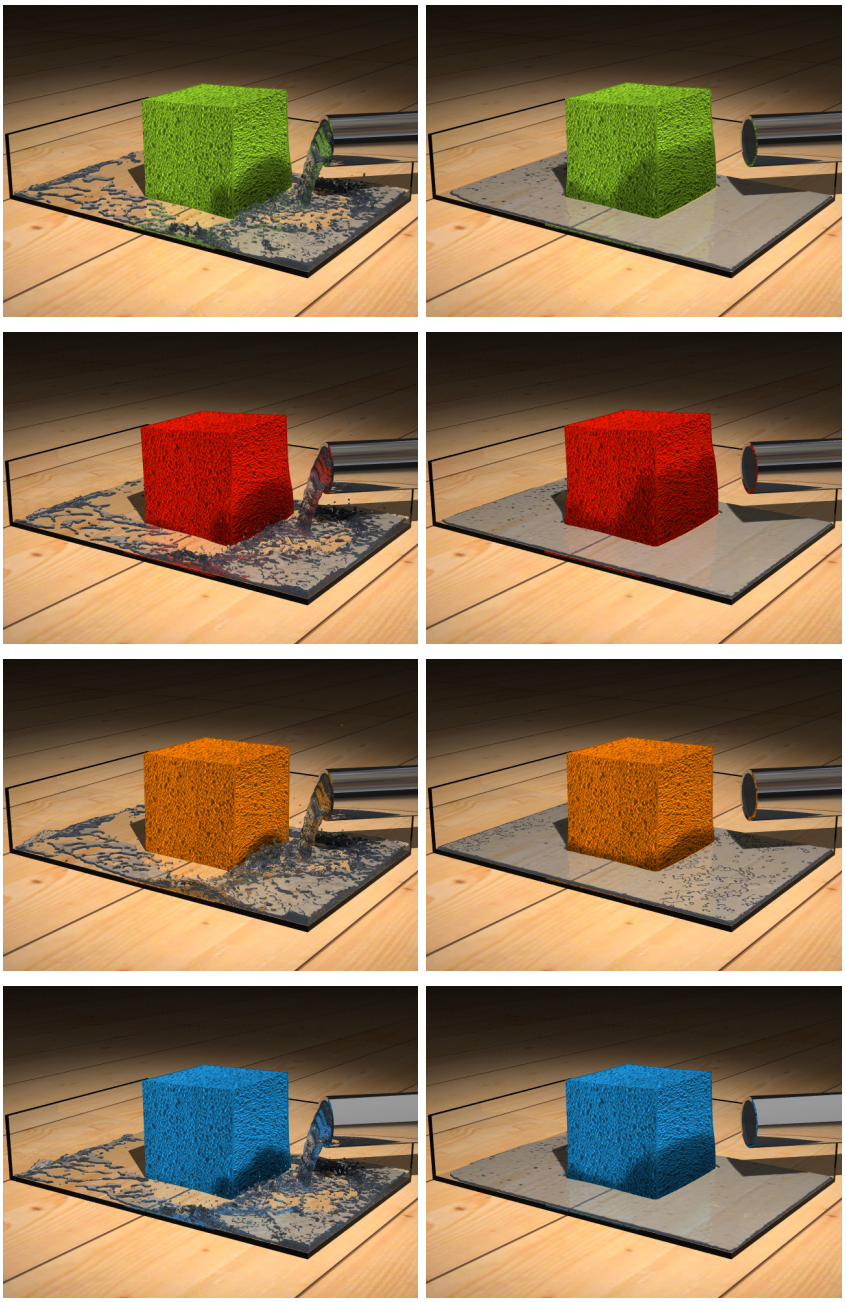


Figure 6.6: Animations showing the effects of the porous flow parameters by pouring water against a dry cubic sponge. Higher pore pressures affect the volume gain (red), a lower permeability slows the flow down (orange) and a low capillary pressure does not pull the water as high (blue).

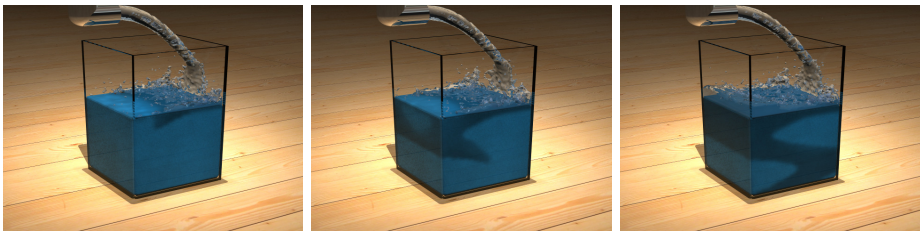


Figure 6.7: Water flowing through a heterogeneous porous material. An S-shaped region has a 500 times higher permeability, allowing the fluid to reach the bottom more easily.

particles is dropped into a funnel. The material is strong enough to support and hold the Bunny above the funnel. As we pour water in the funnel (60,000 particles), the Bunny absorbs the water and becomes weaker. Eventually the material is no longer capable of supporting the increased weight causing the Bunny to sag through the funnel into the reservoir below.

In Figure 6.11(a) we show a dry sand column collapsing as in [Zhu and Bridson, 2005]. Once the sand pile has come to rest we pour water onto the sand. Notice how the water jet mixes with the sand and erodes a hole in the pile. The simulation consists of 45,000 sand and 35,000 water particles and took approximately 50s per animation frame.

Figure 6.12 shows the same sand column, however now the sand is 40% saturated. The surface tension of the present water keeps the column rigidly standing, which is simulated by increasing the cohesion term as in Equation 6.12. A water column is released on the sand column. Notice how the water erodes the lower half of the sand column and weakens the sand structure. Eventually the sand column falls apart in the water. The container is 16cm×10cm×12cm. The simulation consists of 45,000 sand and 75,000 water particles and ran at approximately 78s per frame.

A rain shower is simulated in Figure 6.13. At first only a few drops fall from the sky on the dry soil and can be absorbed rapidly. As more rain drops fall down, the soil becomes saturated, cannot handle the excess water and turns into a puddle of mud. The soil is simulated using $40,000$ particles, the rain consisted of $50,000$ particles. The average computation time was 1 minute per frame.

In Figure 6.11(b) we show dry sand sliding on the Stanford Bunny. Then, we start pouring water on the sliding sand which transforms into more rigid moist sand. Notice how moist sand piles up at the head and right next to the Bunny. In the end of the animation the excess water is also absorbed, weakening the pile and making it collapse. We use the saturation level of the sand particles to guide the stickiness [Clavet et al., 2005] between the surface of the Bunny and

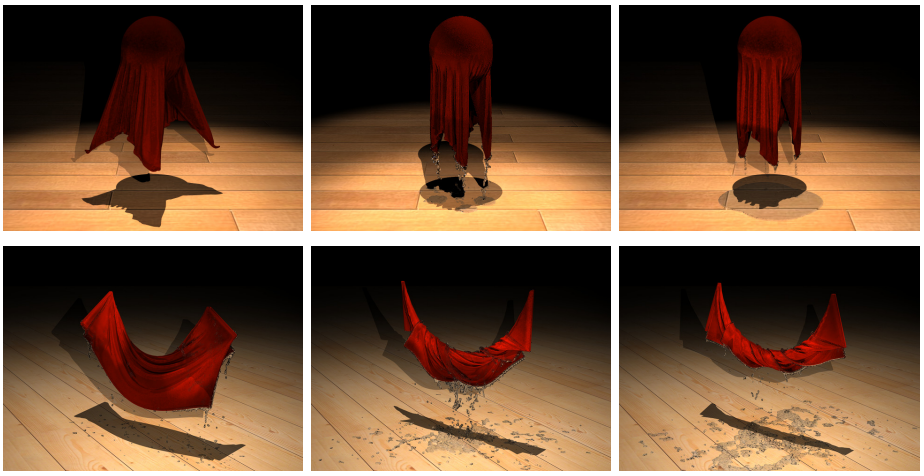


Figure 6.8: Wet cloth simulations. Top: A wet cloth is draped over a sphere. Water flows down the cloth and forms jets at the tips. Bottom: Twisting a wet cloth causes the fluid to flow out of the reduced pore space.

the sand. The Bunny statuette is 20cm×15cm×17cm. Approximately 70,000 particles were used in total, requiring about 1 min of computation per frame.

Timings

Not all simulations were executed on the same hardware and computed using the same software. Results from [Lenaerts et al., 2008] were generated on a single processor computer with a 2.93 GHz CPU and 4 GB of memory. The results featuring granular materials from [Lenaerts and Dutré, 2009b] were generated by an updated and parallelized framework using OpenMP and ran on a quad-core 2.66 GHz CPU and 4 GB of memory. Table 6.1 lists the details about the simulations. Rendering times are not included in the timings.

6.9 Discussion

Our porous flow model is based on a macroscopic approach in which porous particles represent a volume of solid mass and empty space. The most important advantage of using SPH to model the diffusion process is that we can easily handle absorption and emission at the boundary between the fluid and the deformable body. Porous particles and water particles are treated in a uniform manner which greatly simplifies the algorithm. This advantage will be lost when resorting to a mesh-based method for the diffusion process. Modeling the

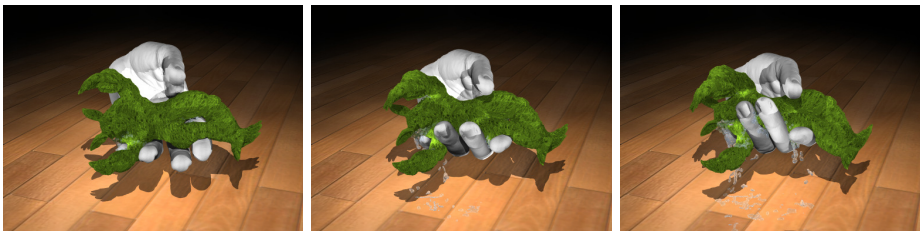


Figure 6.9: A wet porous Armadillo model is squeezed in a hand. The pressure on the pores pushes the fluid outwards.

Simulation	Particles	Step (s)	Frame (s)	CPU (GHz)
Heterogeneity	32,000	0.75	120	1x2.93
Wet Cloth	20,000	0.5	90	1x2.93
Twisting Cloth	20,000	0.5	90	1x2.93
Armadillo	50,000	2	1200	1x2.93
Funnel Bunny	84,000	3.5	300	1x2.93
Jet on Sand	80,000	0.23	50	4x2.66
Sand Column	120,000	1.22	78	4x2.66
Rain	90,000	0.54	60	4x2.66
Sand on Bunny	70,000	0.6	60	4x2.66

Table 6.1: An overview of timing statistics for the presented simulations. Average timings are given per time step and per animation frame.

internal diffusion process using mesh-based approaches should be easy, however, handling the absorption and emission at the interfaces seems to be a non-trivial problem.

In this particle representation fluid can flow from one particle to the next, but flow inside one particle is lost in the representation. We address this by using the saturation level as a scaling factor in Equation 6.8. This effectively stalls flow in the porous particle volume and hereby achieves the desired wet flow front instead of a very gradual diffusion. Using the equation of state for pore pressure calculations holds the same compressibility issues as for SPH fluid simulations. Therefore we cannot guarantee volume preservation of the fluid in a deformable porous body, causing porous flow to start too late when squeezing a porous body using a low pore pressure constant.

Next to the effective stress one could also adjust the elastic parameters to model the behavioral changes in the wet material. While Young's modulus generally seems to decrease with increasing water saturation, the behavior of Poisson's ratio seems less clear. Since these adjustments are highly material

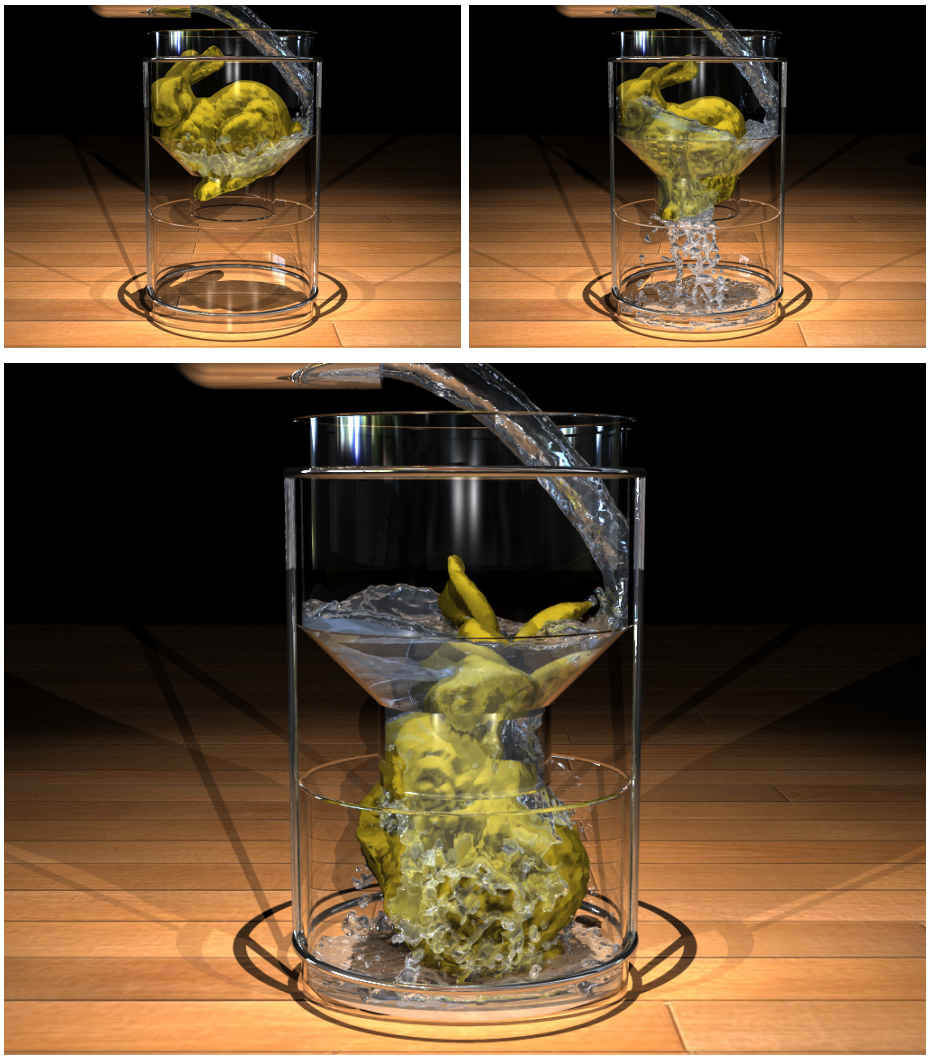
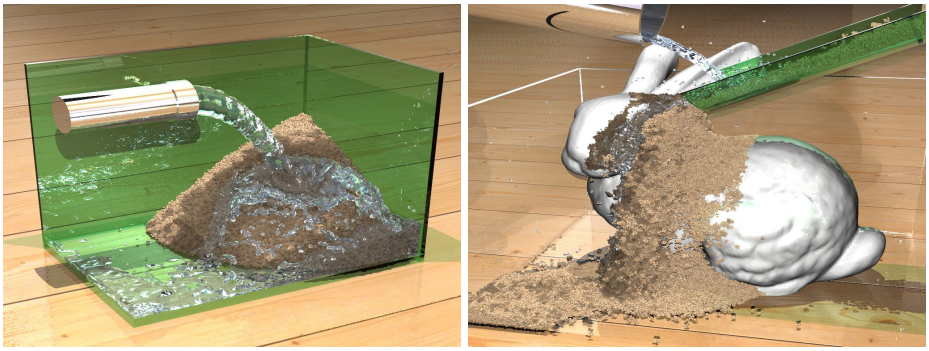


Figure 6.10: A dry porous Stanford Bunny is dropped in a funnel. As water is absorbed the porous material weakens and the Bunny sags through the funnel.



(a) Static Sand Mixing (b) Dynamic Sand Mixing

Figure 6.11: (a) Water is poured on a pile of dry sand. The water percolates into the sand volume and erodes a hole in the pile. (b) Dry sand slides down on a Stanford Bunny. At the same time water is poured on the sand. Moist sand piles up rigidly while wet sand turns in to viscous mud.

Figure 6.12: A moist sand column stands rigidly until a column of water is released. The water percolates through the sand and erodes pieces of the structure.

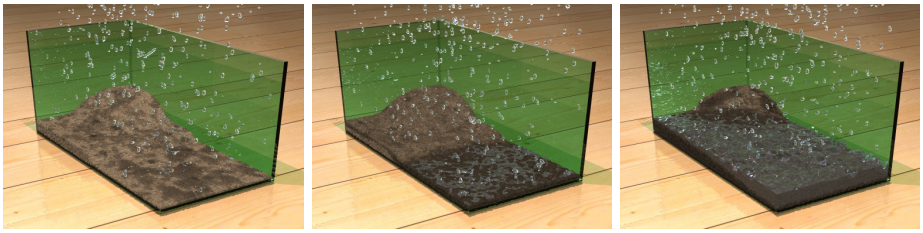


Figure 6.13: A rain shower on dry soil. As the soil becomes saturated a layer of water forms on top and the soil turns to mud.

dependent, we propose to use only the effective stress, which should suffice in most computer graphics applications.

The inclusion of porous flow through bodies requires an extra iteration step over the particles and their neighbors to diffuse the absorbed fluid mass. The capillary and pore pressure and their gradients can be computed in the same steps as the density and density force field respectively [Müller et al., 2003]. However, to compute the pore-velocity field the pressure gradients must be known. This extra iteration over the particles only marginally increases the simulation time. The simulation in the top row of Figure 6.8 took about 9% longer in simulation time for the porous cloth compared to the solid cloth (not accounting for the created fluid particles).

In our experience, the shrinking fluid particles can be the main cause of a slow down of the simulation time. Their smoothing length determines the time step and hence the simulation time per frame. In a single simulation time step, most of the computation time is spent on neighborhood queries (as also noted in [Keiser et al., 2006]). We currently use a grid-based acceleration structure similar to Müller et al. [2003] that works reasonably well. Even though our algorithm introduces differently sized particles, most particles have a uniform size and only a small fraction of particles (typically near the boundary of the porous material) have smaller radii. Using a uniform grid, with grid size based on the largest, most common, support radius therefore only incurs little overhead for the smaller particles. However, we plan to investigate whether a k-d tree would further decrease the time spent on range queries.

For porous objects that only deform a little, the precomputed reference neighborhood can be used to speed up internal diffusion. However, when simulating cloth as in the wringing cloth demo (Figure 6.8), this would not yield the correct results, as pieces of cloth overlap and touch and particle neighborhoods change significantly during simulation. Here, dynamic neighborhood computation results in the correct diffusion behavior.

Also, the effective stress, which is the stress reduced by the absorbed fluid, may cause fractures in the sand volume by increasing particle spacing. These fractures may be influenced by the initial particle sampling, especially for low resolution samplings. Since the porous flow algorithm requires the whole volume to be sampled with porous particles, we cannot use adaptive sampling schemes as in the framework of Pauly et al. [2005]. The techniques of [Desbrun and Cani, 1999; Adams et al., 2007] might be better suited.

In concurrent work, Rungjiratananon et al. [2008] presented an alternative framework for animating the interaction between sand and water. They couple an SPH fluid system to a Discrete Element Method (DEM) for the simulation of granular materials. Fluid percolation is modeled by transferring wetness values and applying a cohesion force between sand particles accordingly. Although they achieve similar effects, the DEM simulates one sand grain by one particle and therefore is subject to the same scalability limitations as the model of Bell et al.

[2005]. In contrast, we simulate the sand volume as a fluid which not only decouples particle resolution from the sand grains, but also facilitates a smooth transition from dry sand to moist sand to mud.
Their method achieves interactive rates by implementing the algorithm on the GPU. In principle our fluid and granular material simulation framework could be accelerated using the same GPGPU technique. However, the porous flow framework creates multi-resolution fluid particles which may hinder such a GPGPU implementation (e.g. typically nearest-neighbors are searched on a grid with only a limited amount of particles per cell).

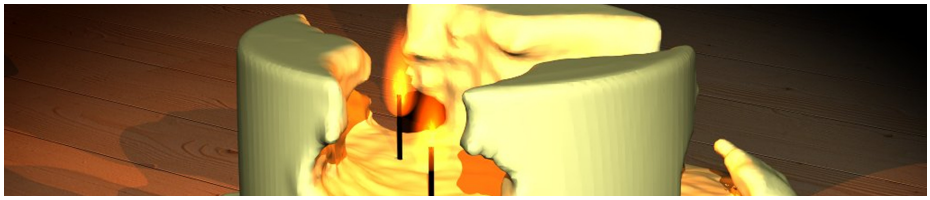
6.10 Conclusion

In this chapter we have presented an SPH method for simulating fluid flowing through a porous material. Rigid and deformable objects and granular volumes are sampled by particles which represent local porosity and permeability distributions at a macroscopic scale. Our diffusion method for porous flow can easily be incorporated into existing particle-based simulations. Also the secondary effects of absorbed fluid on the body can be simulated by slight adjustments to the implemented physics models. Various animations demonstrate these new effects due to absorption, such as material weakening, rigid sand structures and dirt turning into mud, which are not possible with current fluid animation systems.

In future work we would like to extend the porous flow framework to handle multi-phase flow through a porous material. This would allow for simulations with e.g. air-filled pores, which can form bubbles under water. Other examples include interaction with soap and foam or transferring dirt or soil.

Chapter 7

An Architecture for Unified SPH Simulations



Software architecture is an important aspect of computer science. When building an elaborate simulation system as ours, featuring simulation algorithms for fluids, elastics, cloth, rigids and sand, a good architecture is necessary. In fact, we discovered a good architecture can even provide means to extend the range of possible simulations and provide the animator with a tool to produce animations that otherwise would need several animation and simulation packages.

Based on [Lenaerts and Dutré, 2009a] this chapter discusses such an architecture we designed and applied to our existing simulation system. The potential and advantages are illustrated with various animations.

7.1 Introduction

In the last two decades, many techniques to simulate the physical behavior of fluids and solids in computer graphics have been developed and are incorporated in animation tools. They are used to produce various effects in computer animations and movies including characters arising from water, freezing and falling apart, or turning into sand or dust. However such visual effects are currently not possible within a single simulation system. Instead, various simulation and animation packages need to be combined to get the desired result, requiring many adjustments and iterations to tune the inputs and outputs of the pipeline.

The goal of this paper is to extend the physics-based systems to interactively model and simulate transitions from one material to another — or combinations

of several materials and forces — together with a control system to produce attractive visual effects. Reducing the simulation pipeline can shorten the time spent on creating physics-based animations, and provide additional freedom to the animator.

In recent years, several authors have proposed particle-based systems which unify simulation algorithms for fluid flow, rigid body motion and deformations of elastic bodies [Keiser et al., 2005; Solenthaler et al., 2006, 2007; Lenaerts et al., 2008; Lenaerts and Dutré, 2008b]. The main idea consists of using the same solver, usually Smoothed Particle Hydrodynamics (SPH), to compute the necessary equations. Interactions and phase changes are now much easier to simulate since computations are done on the same simulation elements, i.e. particles. The phase changes in existing unified frameworks interpolate between elastic or rigid materials and fluids for melting and solidification. However they are often hard-coded in the simulator and thus limit the possibilities or extensions to other transitions between materials.

We extend existing unified SPH systems to handle continuous changes in physical properties and forces applied on the particles, above and beyond the current capabilities for interactions and phase changes. Therefore, we propose a general system architecture for unifying SPH frameworks where the main idea is to separate particles from the applied forces. Furthermore we implement our architecture into an interactive simulation and modeling prototype application using the latest graphics hardware technology. Based on physical building blocks, this system provides mechanisms to interpolate the behavior of fluids and objects to easily and interactively create animations featuring both physical plausible simulations and special effects.

Overview

We start a detailed discussing of our architecture in Section 7.2. Our implementation is explained in Section 7.3. The potential of our final unified SPH framework is illustrated by various animations in Section 7.4 and discussed in Section 7.5. The final Section 7.6 concludes this chapter.

7.2 Architecture

During a simulation, particular force models are applied on particles. One of the benefits of solving these models in a unified SPH system is that common parts only have to be performed once and similar parts can be grouped together thereby boosting the overall performance. However, instead of only restructuring the way the forces are computed (i.e. grouping similar parts of the solvers), we also propose to restructure the way the forces are applied to provide mechanisms for combining forces and transitions between forces. The key idea is to make an

explicit separation between particles, their properties and applied forces.

As described in Chapter 2, particles are actually points in space with certain properties describing the volume of the simulated fluid or object. During the simulation, particular forces are applied on the points. The main idea of this simulation architecture is to make an explicit separation between particles (points) and their properties and applied forces.

7.2.1 Force Behaviors

Each force has its own set of properties defining the behavior or the characteristics of the force the simulator should apply to the particles. Gravitation is a simple example of such a force, where the constant gravitational acceleration **g** is a property. Other forces, such as wind, explosions or control forces can also be described. We implemented the physics models and their parameters from Chapters 3 to 5 in our system. For example, an elasticity force has properties such as Young's modulus, Poisson's ratio and damping, which are needed to solve Hooke's law (see Figure 7.1).

Instead of applying just one force to a set of particles, the user is free to apply several forces (Figure 7.2). We use the animation time line to define multiple forces. For example, fluid forces together with gravity and control forces. Dimensions different from the time line such as temperature or pressure can also be used. This means forces can be assigned to the whole particle volume (see for example Figure 7.5) or even per particle (see for example Figure 7.8).

Forces can be combined using key frames (Figure 7.2). Each key frame has an associated weight varying from 0 to 1 for the corresponding force. This allows the user to model various transitions and combinations of forces. For each time step, the animator combines the properties of the forces to one set of properties for each particle. Multiple occurrences of the same property are weighted according to the interpolated forces. The ratios of the active forces are also stored in each particle and updated in each time step for use in the simulator.

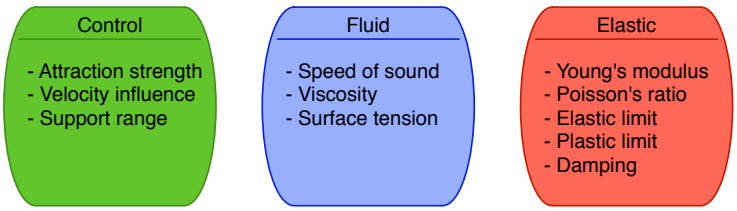


Figure 7.1: Forces on particles, such as fluid or elastic forces, are described by a number of properties characterizing the force the simulator should apply.

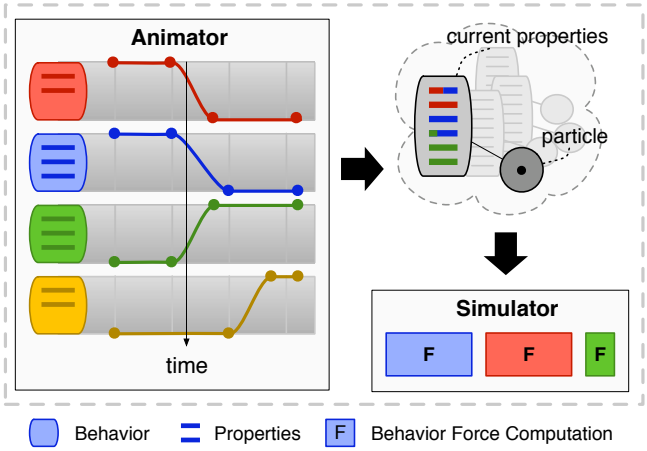


Figure 7.2: The general flow of the architecture. The user can define several forces (i.e. gravity, material forces, ...) on the animation time line using keyframes. In each time step the active forces are evaluated and the resulting properties are stored in the particles. Forces are applied to each particle using the weights defined on the animation time line.

The simulator processes the particles and applies the forces in the ratios as defined by the keyframe weights. Including the force weight in the force computations can sometimes enrich transitions between forces. For example, sand computations can use the weight to control the amount of shearing sand flow versus rigidity.

Defining forces for a certain object or fluid volume can be done in a graphical user interface similar to audio and video editors. The user selects and drags the desired forces on a time line having multiple tracks as in Figure 7.2. They can adjust the default force properties, then set the keyframes and transitions and finally activates the simulator.

7.2.2 Forces

To accommodate the introduced force weighting scheme, a few changes to the classical SPH simulation algorithm are necessary.

Before an actual simulation step can be performed the current state of particle properties must be computed for each particle to know its properties and force weights. Then, an SPH fluid solver typically requires two passes over the particles; one to compute the density for each particle and one to compute forces between particles and advect the particles with the flow. When incorporating rigid, elastic or sand solvers, the last pass is usually split into two or three passes.

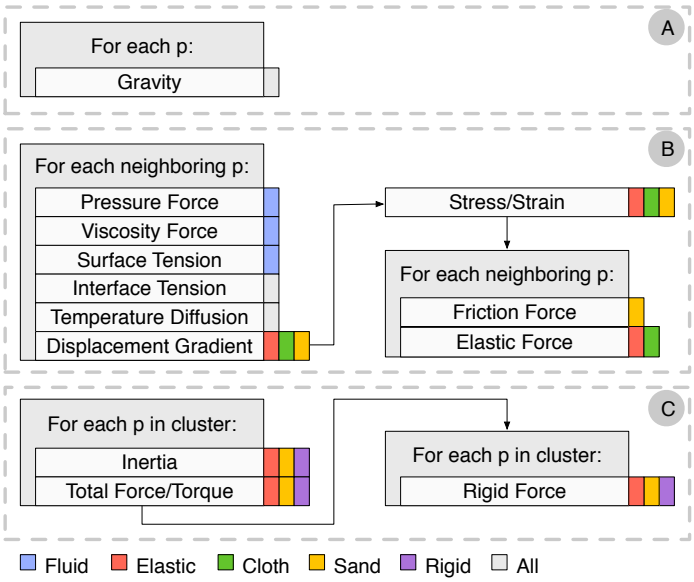


Figure 7.3: The force computations structured by the type of operation on the particles p; (a) Forces acting on single particles, (b) forces acting between neighboring particles and (c) forces acting on a cluster of particles.

For example, to enforce rigid body motion, particle forces need to be computed and accumulated first in order to know the total force and torque on the particle volume.

In general, we see three types of force computations on particles (Figure 7.3). While some forces act on single particles, e.g. gravity, others are applied between neighboring particles (e.g. elasticity) or on a cluster of particles (e.g. rigidity). The key idea is to divide the force computation into three stages corresponding to the aforementioned types. Forces requiring several passes over the particles can be split into subparts. This way, a plug-in architecture arises, making the integration of future forces fairly easy. Figure 7.3 provides a high level overview of the force computations needed to simulate the physics models (forces) discussed in Chapters 3, 4 and 5. The different force computations are grouped into the three types of forces, requiring five passes over the particles. Of course, only the active forces are actually computed.

7.2.3 Transitions and Combinations

We used a linear interpolation scheme to determine the force properties and weights, but any other interpolation scheme can be used.

Our framework generalizes transitions and lets the animator decide the behavior of the transition. Using a combination of forces the desired transition can be modeled. For example, transitioning from or to elastic is done instantaneously by discarding or storing the reference neighborhood, but a highly viscous fluid can be used to provide the illusion of a continuous transition. Likewise, making a fluid rigid, can be done instantaneous, or via a viscous fluid or an elastic material.

As an example of non-physics based forces, we implemented the concept of control particles, presented by Thürey et al. [2006], as a force. Particles with control forces will attract neighboring particles and move them along. However, in [Thürey et al., 2006] the control particles did not take up physical volume. As noted earlier, including the weights in the force computations can be beneficial. We use the force weights to exclude certain particle interactions as a way of dealing with control particles.
Typically, control particles are samples from an animation (see for example Figure 7.7). However, by combining a control force with another force, such as cloth (Figure 7.9), we can perform both simulations at the same time instead of using a pipeline consisting of several simulation tools.

7.3 System Implementation

The proposed architecture was first implemented by refactoring our existing SPH system for fluid flow and elastic simulations on the CPU. This proved to be fairly straightforward, illustrating the simplicity of the proposed architecture. Second, a new GPU simulation tool was built in which the primary objective was modeling and simulating at interactive frame rates. Being able to model the transitions and combinations of behaviors, and checking the effects on the simulation interactively, is an important step forward for animators.

Resembling a typical 3D modeling package the user is presented an OpenGL rendering of the scene containing the particle volume (Figure 7.4). A secondary window showing the keyframed behaviors over a time line is available. Here, the user can modify the keyframes. Simulation parameters and force properties can be edited in the settings window on the right. Other modeling functionalities include pausing and playing the animation, rewinding the simulation to a certain point in time, adjusting keyframes and resuming the simulation from that point on.

In Figure 7.4 we show a live screen capture while we model a breaking dam simulation which assumes the shape of a rubber ducky using control particles. Timing the control force versus the water simulation can be a labor intensive, trial-and-error process on today's frameworks. Running at 25% of real-time simulations however, this interactive simulation allows for rapid refinements of timings.

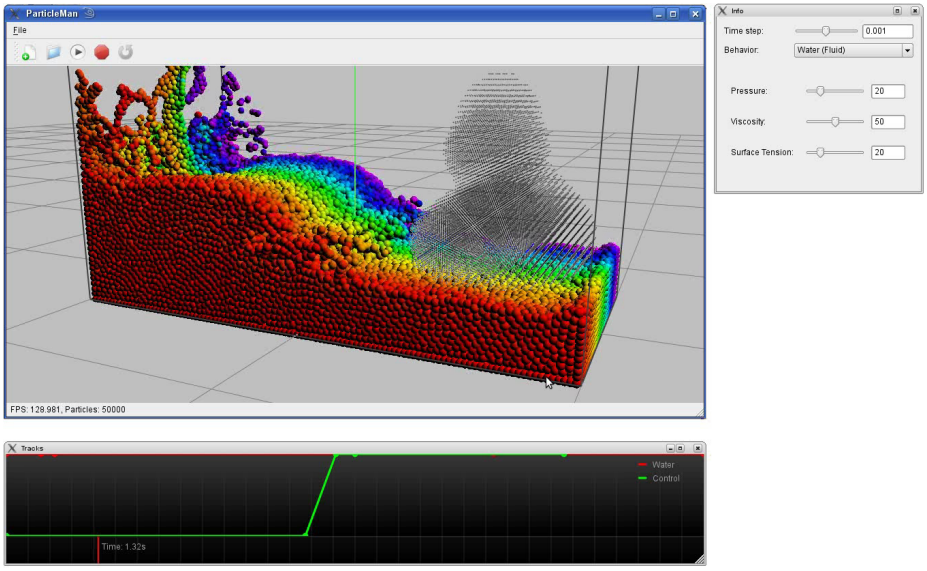


Figure 7.4: Our interactive prototype application showing an OpenGL rendering of the particle volume and the keyframes on the behavior track. Interactive simulations are achieved by implementing the simulation engine using CUDA.

Our prototype modeling tool was implemented using NVIDIA's CUDA technology [NVIDIA, 2006] allowing for parallel execution of the force computations on the particle volume. Compared to a fluid simulation performed on a single-core CPU, which needs $0.5s$ per time step to process $50,000$ particles, our CUDA implementation is 125 times faster running at an average of 250 time steps per second on a 2.93 GHz CPU with 4 GB of memory and a GeForce GTX 280 GPU. Although this simulation engine is not fully optimized, it does illustrate the potential of SPH simulations on graphics hardware and the prospects of our modeling tool.

7.4 Animations

We modeled a set of animations illustrating the potential of our framework (see also the accompanying videos). While some of these animations may be possible within existing frameworks, our system allows them to be simulated in one framework with much more control for the animator.

To illustrate changing particle properties over time, we modeled three fluids with different densities put on top of each other (see Figure 7.5). Once the fluids

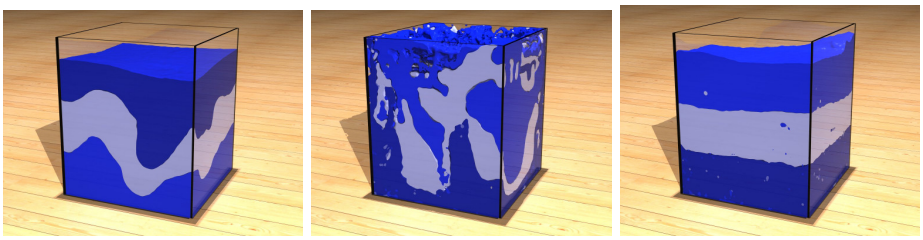


Figure 7.5: Three fluids with different densities are layered in equilibrium. When we change the density of the bottom and top fluids, Rayleigh-Taylor instability is shown and the two fluids switch places.

Figure 7.6: A red liquid is poured into the shape of a rubber ducky, which then hardens to an elastic material and falls down into a container of water.

come to rest, we gradually change the density of the top and bottom fluids. As a result, both fluids switch places. The simulation was performed on $260,000$ particles taking approximately 4 minutes per animation frame on a 2.66GHz quad-core computer.

Animating the transition from a fluid to an elastic body is performed in Figure 7.6. A red fluid is poured into the shape of a rubber ducky. Once the shape is completely filled the fluid hardens and is dropped in the water container below. There, the less dense ducky floats on the water until we change it back to fluid. The simulation consists of 230,000 particles, taking on average $40s$ to compute one animation frame or $0.60s$ for each step on an 3.0GHz 8-core computer.

In Figure 7.7 a galloping horse animation guides a simulation using control particles. During the animation the simulated volume seamlessly fades from sand to water and from water to honey. Notice the flow differences for each of the substances. While water nicely follows the horse movements, the highly viscous honey and the friction in the sand try to resist the flow. Approximately 170,000 particles were processed simulating sand, water and honey, taking $1.5s$ for each time step or $70s$ per animation frame on a 3.0GHz 8-core computer.

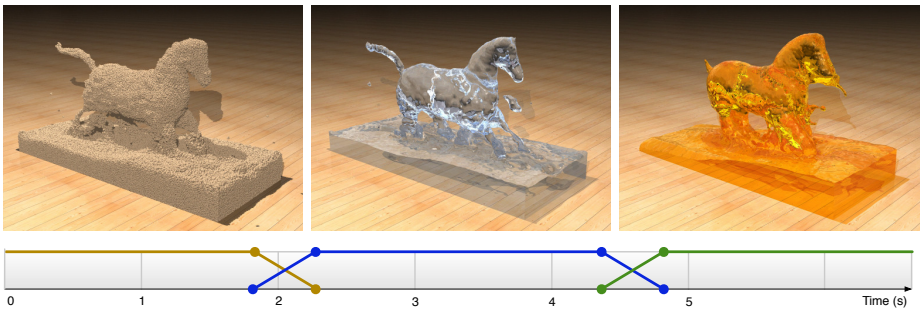


Figure 7.7: A galloping horse animation guides a simulation seamlessly fading between sand, water and honey. The keyframes defining the transitions between the three forces are visualized on the animation time line below.

Figure 7.8: A burning candle simulation showing melting and re-solidifying of wax. The particle's temperature is used to fade between the elastic and liquid regime.

Figure 7.8 is an example of defining keyframes for forces on other dimensions than time. We modeled a burning candle showing melting an re-solidifying of wax. This example applies the capabilities of the proposed architecture in a more realistic setting. The particle's temperature controls the transition from elastic material to fluid (melting) and back (re-solidifying). Compared to other unified SPH frameworks, such as [Solenthaler et al., 2006], which also uses an interpolation between elastic and fluid for melting, our scheme allows the animator to model the precise transitions in a graphical user interface instead of hard-coding the interpolation. Spatially varying properties is one of the advantages of particle-based approaches and our scheme further expands this to spatially varying forces. The candle consists of 20,000 particles requiring $0.15s$ per time step on a 2.66GHz quad-core computer.

As discussed in Section 7.2.3, control particles can be steered by a simulation instead of a predefined animation sequence. In Figure 7.9 a character performs a

Simulation	Particles	Step (s)	Frame (s)	CPU (GHz)
Densities	260,000	0.70	240	4x2.66
Ducky	230,000	0.60	40	8x3.0
Horse	170,000	1.50	70	8x3.0
Candle	20,000	0.15	27	4x2.66
Dress	136,000	0.49	35	4x2.66

Table 7.1: An overview of timing statistics for the presented simulations. Average timings are given per time step and per animation frame.

dance wearing a dress made of red paint. We use the cloth simulator to provide the dynamics of control particles which sample the dress. Extra fluid is emitted over the entire surface of the dress to keep the dress sampled with fluid despite the splashes and gravity. The simulation started with approximately 18,000 cloth/control particles and 18,000 fluid particles. By the end of the animation 100,000 additional fluid particles were created.

An overview of the particle numbers and CPU timings of the discussed simulations is given in Table 7.1. Figure 7.5 required more than twice the simulation time compared to the other simulations because we used a smaller time step to keep the large density differences stable. The horse animation (Figure 7.7) consisted of approximately 6,000 control particles for each animation frame. This required additional nearest neighbor queries with double support ranges to shape the flow, which explains the longer simulation times. Overall, we did not notice any significant slow down in simulation times compared to the framework without the introduced behavior scheme.

7.5 Discussion

We chose to work with an SPH framework unifying the solvers for fluids, rigid and elastic bodies, cloth and sand. This may not be the best framework to simulate certain materials. For example, a better choice to simulate elasticity are Moving Least Squares techniques [Müller et al., 2004a] which are rotation invariant. However, the benefit is that the different SPH solvers can be integrated into one framework much easier. As illustrated by Figure 7.3 in Section 7.2.2 similar parts can be found and integrated, which also benefits performance. Our architecture introduces some extra work since particle properties have to be computed from the key-framed forces in each time step, but this is only a marginal cost compared to the nearest neighbor queries and force computations needed for an SPH solver.

The force weighting scheme provides freedom to apply forces and transitions between materials. However, it could be expanded by introducing separate

Figure 7.9: A dancing woman wearing a dress made of red paint. The paint splashes as the dress sweeps around. Simultaneously simulating cloth dynamics and fluid flow is possible by combining control forces and elastic forces.

weights or interpolation schemes per property to provide more detailed options.

Surface meshes were extracted using the color field of [Solenthaler et al., 2007] presented in Chapter 2 and the marching cubes algorithm [Lorensen and Cline, 1987]. Often, a high resolution mesh is animated by sampling the volume with particles and matching the mesh to the particles' displacements [Müller et al., 2004a]. For large deformations, such as transitions to fluids, the deformed mesh should comply with topology changes of the underlying particle model, such as merging and splitting, which is not trivial. In our experiments, we found a simple blend between the two meshes during rendering to suffice. However, a better solution was recently presented by Wojtan et al. [2009].

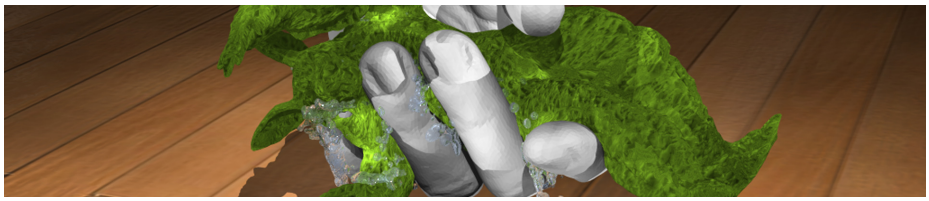
7.6 Conclusion

We have presented a general architecture for unifying SPH frameworks featuring an efficient plug-in structure for force computations. The main idea of the architecture is separating particles from the applied forces in order to easily key-frame the simulation. We implemented our architecture as an animation and simulation tool using CUDA. The result is an interactive system based on physical and non physical components where various forces can be combined and transitions between materials and forces can be modeled. We showed animations showing physically plausible melting and solidifying behavior, but also fictional transitions from sand to water or fluids behaving as cloth.

Though the presented animations were modeled using an OpenGL preview window and a basic graphical user interface to define behaviors and keyframes, this did not prevent us showing various special effects. Integrated into a proper animation modeling package, we believe our approach will be a powerful tool for designing a wide range of animation effects.

Chapter 8

Conclusion



This final chapter summarizes the work presented in this dissertation. The main ideas and our achievements are presented once more in Section 8.1. Contributions are summarized in Section 8.2 and thoughts on future work are written down in Section 8.3.

8.1 Summary

In the last decade fluid simulations have received quite a lot of attention in computer graphics animations. In more recent years fluid animations became more mature and authors have presented the simulation of new phenomena such as the formation of foam [Cleary et al., 2007] or presented new and better ways to simulate interactions [Solenthaler and Pajarola, 2008; Rungjiratananon et al., 2008] and phase changes [Solenthaler et al., 2007]. It is this trend this dissertation has followed. We have focused on interactions in which the fluid can affect the object and its material it is interacting with.

Our main observation of previous fluid animations was the lack of wet surfaces. Fluids only interacted with the (deforming) geometry of objects. In reality however many materials can be defined as porous at a certain scale, meaning many objects can absorb water inside their volume. We presented a novel algorithm capable of simulating this important phenomenon. The center piece of this algorithm is the concept of porous particles which make abstraction of local pore structures. We applied the law of Darcy governing porous flow on these porous particles, making the internal fluid flow possible by means of a diffusion process. We coupled porous objects to external fluids to be able to simulate fluid absorption and emission. Our algorithm treats both the internal flow as well as the

external flow in a uniform manner hereby benefitting from the underlying particle framework. We showed how secondary effects of the absorbed fluid on the porous material can be incorporated in existing particle-based frameworks for simulating rigid bodies, elastic bodies, cloth and granular materials. This lead to the animation of new unseen effects including mass and buoyancy changes, weakening of elastic materials, sticky wet cloth, moist sand sculptures and mud formation.

To accommodate the new porous flow algorithm a large particle framework needed to be built. We not only implemented various existing algorithms for simulating fluid flow, elastic bodies, rigid bodies and granular materials in one unifying framework, but we also created new simulation algorithms for cloth and sand in the same unifying manner.

The simulation systems developed in computer graphics try to relieve animators from having to model real-life phenomena accurately and realistically. Our porous flow contributions were developed with this goal in mind. On the other hand, the simulation systems also need to provide as much freedom for the animator as possible. Therefore we proposed a novel architecture for particle-based simulation systems capable of simulating a wide range of materials. This architecture divides the simulation system into building blocks (i.e. behaviors) and transforms them into true animation tools by which various combinations and transitions can be modeled. We implemented this architecture in a prototype animation system featuring interactive simulation and manipulation of the particle volumes, giving the animator rapid feedback about the animated scene. Our various animations demonstrate both plausible simulations as fictional animations.

The work described in this dissertation mainly contributes in the field of physics-based animations. We were the first to present the full 3D simulation of porous flow and its effects in the domain of computer graphics. We hope we have hereby opened a door to a new line of research or at least motivated others to broaden their scope of applications and incorporate porous materials.

8.2 Summary of Contributions

Below we summarize this dissertation per chapter and the corresponding original contributions.

Chapter 4

We have presented the two-way coupling of a fluid to thin deformable shells in a unified particle model using Smoothed Particle Hydrodynamics (SPH) for the simulation of fluid and shells. Our results show realistic shell and cloth animations interacting with fluids without any leaks.

This work was developed as part of [Lenaerts et al., 2008] and was also presented at ACM SIGGRAPH as a poster [Lenaerts and Dutré, 2008a].

Chapter 5

In this chapter we have shown how the fluid-sand model of Zhu and Bridson [2005] can be simulated in a unified SPH framework. Simulating sand as continuous volumes enabled animations consisting of larger amounts of sand compared to previous approaches. We've also studied the visualization of sand and conclude rendering pseudo-random sand grains for each simulation particle offers a good balance between speed and conveying the illusion of sand. Furthermore it offers a nice flexibility which provides ways for post-process refinements.

This SPH sand simulation algorithm was published in [Lenaerts and Dutré, 2009b] and presented at the Eurographics 2009 conference in Munich, Germany. Porting the original fluid-sand model to SPH was performed by the author. During the experiments for rendering sand volumes the author was assisted by Jef-Aram Van Gorp, a master student.

Chapter 6

This chapter presented the first algorithm for simulating porous flow in computer graphics. Porous objects are sampled by particles which represent local porosity and permeability distributions at a macroscopic scale. Using an SPH fluid method we diffuse fluid mass through these porous particles. Not only the internal flow, but also fluid absorption and emission were described. We showed how our diffusion method for porous flow and secondary effects because of the absorbed fluid can easily be applied to existing particle frameworks for rigid, elastic and granular materials. Various animations demonstrate these effects, such as material weakening, rigid sand structures and dirt turning into mud, which are not possible with previous fluid animation systems.

The porous flow framework presented in this chapter is a collaboration with Bart Adams (Stanford University) and Philip Dutré (Katholieke Universiteit Leuven) [Lenaerts et al., 2008]. It was presented at the annual ACM SIGGRAPH conference in Los Angeles, California. Bart Adams contributed his implementation of distance fields, shape matching and provided the closing hand model. The author constructed and implemented the porous flow algorithm. He also modeled and rendered the accompanying animations. Peter Vangorp and Jurgen Laurijssen assisted during the final rendering stages. In [Lenaerts and Dutré, 2009b] the author extended his porous flow framework and applied it to sand volumes to be able to mix fluids and granular materials.

Chapter 7

We have presented a general architecture for unifying SPH frameworks featuring an efficient plug-in structure for force computations. The main idea of the architecture is to separate particles from the applied forces in order to easily key-frame the simulation. We implemented our architecture as an animation and simulation tool using CUDA. The result is an interactive system based on physical and non-physical components where various forces can be combined and transitions between materials and forces can be modeled. We presented animations showing physically plausible melting and solidifying behavior, but also fictional transitions from sand to water or fluids behaving as cloth.

Though the presented animations were modeled using an OpenGL preview window and a basic graphical user interface to define behaviors and keyframes, this did not prevent us from showing various special effects. Integrated into a proper animation modeling package, we believe our approach will be a powerful tool for designing a wide range of animation effects.

The author was responsible for redesigning his framework into the proposed architecture. He also re-implemented this framework on graphics hardware to achieve an interactive modeling tool. All animations were modeled and rendered by the author. This work is described in Lenaerts and Dutré [2009a].

8.3 Future Work

Research is never finished and our work is no exception either. Here we describe possible extensions and directions for future research related to our work and to the domain of particle-based fluid simulations in general.

- In Chapter 4 we described the simulation of thin shells and cloth in combination with fluids. Though we can handle thin shells containing fluids without leaking, far elastic stretching can still cause large spaces between particles through which fluid particles can move. Adaptively resampling those regions should solve this problem and even make new simulations such as water-filled balloons possible.

- In Chapter 5 we investigated the simulation and rendering of sand volumes. As future work we would like to do more experiments to ameliorate both the simulation as the rendering of sand. First steps in this direction are already being taken by Alduán et al. [2009]. An SPH adaptation of their technique should deserve attention.

- An SPH framework for simulating porous flow was described in Chapter 6. It is however limited to one single phase. Because todays animation systems are also capable of multi-phase simulations, it would be beneficial to extend our porous flow framework to handle such internal multi-phase

flow. In other domains such as geology, research has already been done on multi-phase porous flow, so it should be possible to adapt the necessary physics to our framework. Then, new effects such as a sponge interacting with water and foam or cloth with water and dirt become possible in computer graphics.

- In recent years APIs for GPU programming become more and more versatile and user-friendly. As a consequence fluid simulations on the GPU are now really taking off. This does not only decrease the computation time for fluid simulations, but will also lead to more detailed and complex fluid animations and the simulation of new phenomena. With this in mind, it would be interesting to see our prototype from Chapter 7 being further developed into a full animation package. Also recent advances in GPU implementations of adaptive particle models [Yan et al., 2009] may make a performance increase of our porous flow framework possible.

From personal communications and recent publications we can conclude that our work is being acknowledged and starts to have a certain impact on the computer graphics community [Becker et al., 2009a; Solenthaler and Pajarola, 2009; Krištof et al., 2009; Alduán et al., 2009; Robinson-Mosher et al., 2009; Pabst et al., 2009; Oh et al., 2009; Gao et al., 2009] and even other domains such as geology. We hope this work will inspire new research in the field of computer animation and will lead to new and attractive fluid animations.

List of Publications

This is a list of the scientific publications by the author (2005 – 2009). The list also includes publications not discussed in this dissertation.

Articles in International Reviewed Journals

- **Toon Lenaerts**, Bart Adams, and Philip Dutré. Porous Flow in Particle-Based Fluid Simulations. *ACM Transactions on Graphics*, 27(3), Proceedings of ACM SIGGRAPH 2008, pages 49:1–49:8, 2008.
- **Toon Lenaerts** and Philip Dutré. Mixing Fluids and Granular Materials. *Computer Graphics Forum*, 28(2), Proceedings of Eurographics 2009, pages 213-218, 2009.

Contributions at International Conferences, Published in Proceedings

- See *Articles in International Reviewed Journals*.

Contributions at International Conferences, not Published or Only as Abstract

- **Toon Lenaerts** and Philip Dutré. Unified SPH Model for Fluid-Shell Simulations. ACM SIGGRAPH 2008 Poster. ACM SIGGRAPH 2008, Los Angeles, USA, August 2008.
- Peter Vangorp, Olivier Dumont, **Toon Lenaerts**, and Philip Dutré. A Perceptual Heuristic for Shadow Computation in Photo-realistic Images. ACM SIGGRAPH 2006 Sketch, p. 102. ACM SIGGRAPH 2006, Boston, USA, August 2006.

Technical Reports

- Bart Adams, **Toon Lenaerts**, and Philip Dutré. Particle splatting: Interactive rendering of particle-based simulation data. Report CW 453,

Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, July 2006.

- **Toon Lenaerts** and Philip Dutré. A Unified SPH Model for Fluid-Shell Simulations. Report CW 530, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, November 2008.

- Ares Lagae, Peter Vangorp, **Toon Lenaerts**, and Philip Dutré. Isotropic Stochastic Procedural Textures by Example. Report CW 546, Department of Computer Science, K.U.Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, May 2009.

- **Toon Lenaerts** and Philip Dutré. An Architecture for Unified SPH Simulations. Report CW 559, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, August 2009.

Miscellaneous

- **Toon Lenaerts** and Frederic Taes. Physically Based Modeling and Visualisation of Smog. Master thesis, Katholieke Universiteit Leuven, Departement Computerwetenschappen, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, May 2005.

Biography

Toon Lenaerts was born on May 25, 1983 in Tienen and grew up in Wommersom, Linter. After his teenage years majoring in Latin and mathematics he moved to Leuven where he received a Bachelor's degree in Computer Science (Kandidaat in de Informatica) from the Katholieke Universiteit Leuven in 2003. In 2005 he graduated cum laude as Master in Computer Science (Licentiaat in de Informatica) at the same university with a dissertation on simulating and rendering smog under the supervision of Prof. dr. ir. Philip Dutré and the guidance of Ares Lagae. In October 2005 he shortly joined the research group of Prof. dr. Marie-Francine Moens at ICRI-LIIR where he worked on IBBT projects. In January 2006 Toon returned to his Computer Graphics passion and started his Ph.D. study at the Computer Graphics Research Group of the Katholieke Universiteit Leuven, advised by Prof. dr. ir. Philip Dutré and funded by an IWT scholarship. He completed his Ph.D. thesis in December 2009.

Bibliography

Autodesk Maya. `http://www.autodesk.com`, 2009.

Blender. `http://www.blender.org`, 2009.

POV-Ray: The persistence of vision raytracer. `http://www.povray.org`, 2009.

Adams, B. *Point-Based Modeling, Animation and Rendering of Dynamic Objects*. Ph.D. thesis, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, 2006.

Adams, B., Pauly, M., Keiser, R., and Guibas, L. J. Adaptively sampled particle fluids. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 48. ACM, New York, NY, USA, 2007.

Alduán, I., Tena, Á., and Otaduy, M. A. Simulation of high-resolution granular media. In *Proc. of Congreso Español de Informática Gráfica*. 2009.

Ammann, C., Bloom, D., Cohen, J. M., Courte, J., Flores, L., Hasegawa, S., Kalaitzidis, N., Tornberg, T., Treweek, L., Winter, B., and Yang, C. The birth of sandman. In *SIGGRAPH 2007 sketches*. 2007.

Baraff, D. An introduction to physically based modeling: Rigid body simulation - unconstrained rigid body dynamics. *ACM SIGGRAPH Course Notes*, 1997.

Baraff, D. and Witkin, A. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH 98*, pages 43–54. 1998.

Bargteil, A. W., Sin, F., Michaels, J. E., Goktekin, T. G., and O'Brien, J. F. A texture synthesis method for liquid animations. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 345–351. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006.

Bear, J. *Dynamics of Fluids in Porous Media*. Dover Publications, 1972.

Becker, M., Ihmsen, M., and Teschner, M. Corotated sph for deformable solids. In *Proceedings of the Eurographics Workshop on Natural Phenomena*, pages 27–34. Munich, Germany, 2009a.

Becker, M. and Teschner, M. Weakly compressible SPH for free surface flows. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217. 2007.

Becker, M., Tessendorf, H., and Teschner, M. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 2009b.

Bell, N., Yu, Y., and Mucha, P. J. Particle-based simulation of granular materials. In *Symposium on Computer Animation (SCA 2005)*, pages 77–86. ACM Press, New York, NY, USA, 2005.

Berry, R. A., Martineau, R. C., and Wood, T. R. Particle-based direct numerical simulation of contaminant transport and deposition in porous flow. *Vadose Zone Journal*, 3(1):164–169, 2004.

Carlson, M., Mucha, P. J., and Turk, G. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, 23(3):377–384, 2004.

Chanclou, B., Luciani, A., and Habibi, A. Physical models of loose soils dynamically marked by a moving object. In *CA '96: Proceedings of the Computer Animation*, page 27. IEEE Computer Society, Washington, DC, USA, 1996.

Chentanez, N., Goktekin, T. G., Feldman, B. E., and O'Brien, J. F. Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 83–89. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006.

Chu, N. S.-H. and Tai, C.-L. Moxi: real-time ink dispersion in absorbent paper. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 504–511. ACM Press, New York, NY, USA, 2005.

Clavet, S., Beaudoin, P., and Poulin, P. Particle-based viscoelastic fluid simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 219–228. ACM Press, New York, NY, USA, 2005.

Cleary, P. W., Pyo, S. H., Prakash, M., and Koo, B. K. Bubbling and frothing liquids. *ACM Transactions on Graphics*, 26(3):97, 2007.

Colin, F., Egli, R., and Lin, F. Computing a null divergence velocity field using smoothed particle hydrodynamics. *Journal of Computational Physics*, 217(2):680 – 692, 2006.

Cuesta, C., van Duijn, C., and Hulshof, J. Infiltration in porous media with dynamic capillary pressure: travelling waves. Technical Report MAS-R9932, CWI, 1999.

Darcy, H. *Les Fontaines Publiques de la Ville de Dijon, Dalmont, Paris*. 1856.

Desbrun, M. and Cani, M.-P. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96*, pages 61–76. 1996.

Desbrun, M. and Cani, M.-P. Space-time adaptive simulation of highly deformable substances. Technical Report 3829, INRIA, BP 105 - 78153 Le Chesnay Cedex - France, 1999.

Desbrun, M., Schröder, P., and Barr, A. Interactive animation of structured deformable objects. In *Graphics Interface 99*. 1999.

Falappi, S. and Gallati, M. SPH simulation of water waves generated by granular landslides. In *Proceedings of the 32nd Congress of IAHR*. 2007.

Ferréol, B. and Rothman, D. H. Lattice-boltzmann simulations of flow through fontainebleau sandstone. *Transport in Porous Media*, 20(1–2):3–20, 1995.

Foster, N. and Fedkiw, R. Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30. ACM, New York, NY, USA, 2001.

Foster, N. and Metaxas, D. Realistic animation of liquids. *Graph. Models Image Process.*, 58(5):471–483, 1996.

Foster, N. and Metaxas, D. Controlling fluid animation. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International*, page 178. IEEE Computer Society, Washington, DC, USA, 1997.

Gao, Y., Li, C.-F., Hu, S.-M., and Barsky, B. A. Simulating gaseous fluids with low and high speeds. *Computer Graphics Forum, Special issue of Pacific Graphics 2009*, 28(7):1845–1852, 2009.

Gingold, R. A. and Monaghan, J. J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. astr. Soc.*, 181:375–389, 1977.

Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. Coupling water and smoke to thin deformable and rigid shells. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 973–981. ACM Press, New York, NY, USA, 2005.

Harada, T., Koshizuka, S., and Kawaguchi, Y. Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International*, pages 63–70. 2007.

Hegeman, K., Carr, N. A., and Miller, G. S. Particle-based fluid simulation on the GPU. In Vassil N. Alexandrov, Geert Dick van Albada, P. M. S. and Dongarra, J., editors, *Computational Science – ICCS 2006*, volume 3994 of *LNCS*, pages 228–235. Springer, 2006.

Higuera, F. J. and Jiménez, J. Boltzmann approach to lattice gas simulations. *Europhysics Letters*, 9:663–+, 1989.

Higuera, F. J., Succi, S., and Benzi, R. Lattice gas dynamics with enhanced collisions. *Europhysics Letters*, 9:345–+, 1989.

Hilfer, R. Transport and relaxation phenomena in porous media. *Advances in Chemical Physics*, XCII:299, 1996.

Hilfer, R. Macroscopic capillarity and hysteresis for flow in porous media. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 73(1):016307, 2006.

Hong, J.-M., Lee, H.-Y., Yoon, J.-C., and Kim, C.-H. Bubbles alive. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–4. ACM, New York, NY, USA, 2008a.

Hong, W., House, D., and Keyser, J. Adaptive particles for incompressible fluid simulation. *The Visual Computer (Proceedings of Computer Graphics International)*, 24(7-9):535–543, 2008b.

Jensen, H. W., Legakis, J., and Dorsey, J. Rendering of wet materials. In *Rendering Techniques, Proceedings of the Eurographics Workshop on Rendering*, pages 273–282. 1999.

Keiser, R., Adams, B., Dutré, P., Guibas, L. J., and Pauly, M. Multiresolution particle-based fluids. Technical report, ETH Zurich, 2006.

Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutré, P., and Gross, M. A unified Lagrangian approach to solid-fluid animation. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, pages 125–133. 2005.

Kipfer, P. and Westermann, R. Realistic and interactive simulation of rivers. In Mann, S. and Gutwin, C., editors, *Proceedings Graphics Interface 2006*, pages 41–48. Canadian Human-Computer Communications Society, 2006.

Kolb, A. and Cuntz, N. Dynamic particle coupling for GPU-based fluid simulation. In *Proc. 18th Symposium on Simulation Technique*, pages 722–727. 2005.

Krištof, P., Beneš, B., Křivánek, J., and Šťava, O. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum (Proceedings of Eurographics 2009)*, 28(2), 2009.

Kwatra, V., Adalsteinsson, D., Kim, T., Kwatra, N., Carlson, M., and Lin, M. Texturing fluids. *IEEE transactions on visualization and computer graphics*, 13(3), 2007.

Lee, H.-Y., Hong, J.-M., and Kim, C.-H. Interchangeable SPH and level set method in multiphase fluids. *The Visual Computer*, 25(5):713–718, 2009.

Lenaerts, T., Adams, B., and Dutré, P. Porous flow in particle-based fluid simulations. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2008)*, 27(3):49:1–49:8, 2008.

Lenaerts, T. and Dutré, P. Unified SPH model for fluid-shell simulations, 2008a. ACM SIGGRAPH 2008 Poster.

Lenaerts, T. and Dutré, P. Unified SPH model for fluid-shell simulations. Technical Report 530, Katholieke Universiteit Leuven, 2008b.

Lenaerts, T. and Dutré, P. An architecture for unified SPH simulations. Technical Report CW559, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, 2009a.

Lenaerts, T. and Dutré, P. Mixing fluids and granular materials. *Computer Graphics Forum*, 28(2):213–218, 2009b.

Li, X. and Moshell, J. M. Modeling soil: Realtime dynamic models for soil slippage and manipulation. In *In Computer Graphics Proceedings, Annual Conference Series*, pages 361–368. 1993.

Liu, G. R. and Liu, M. B. *Smoothed Particle Hydrodynamics: A meshfree particle method*. World Scientific, 2003.

Lorensen, W. E. and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, New York, NY, USA, 1987.

Losasso, F., Talton, J., Kwatra, N., and Fedkiw, R. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.

Lucy, L. B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

Manwart, C., Aaltosalmi, U., Koponen, A., Hilfer, R., and Timonen, J. Lattice-boltzmann and finite-difference simulations for the permeability for three-dimensional porous media. *Phys. Rev. E*, 66(1):016702, 2002.

Martys, N. S. and Chen, H. Simulation of multicomponent fluids in complex three-dimensional geometries by the lattice boltzmann method. *Phys. Rev. E*, 53(1):743–750, 1996.

Mihalef, V., Metaxas, D., and Mark, S. Textured liquids based on the marker level set. *Computer Graphics Forum*, 26(3):457–466, 2007.

Miller, G. and Pearce, A. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

Monaghan, J. J. Smoothed Particle Hydrodynamics. *Annual review of astronomy and astrophysics*, 30(A93-25826 09-90):543–574, 1992.

Monaghan, J. J. Simulating free surface flows with sph. *Journal of Computational Physics*, 110(2):399–406, 1994.

Monaghan, J. J. Smoothed Particle Hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, 2005.

Morris, J. P., Fox, P. J., and Zhu, Y. Modeling low reynolds number incompressible flows using SPH. *J. Comput. Phys.*, 136(1):214–226, 1997.

Morris, J. P., Zhu, Y., and Fox, P. J. Parallel simulations of pore-scale flow through porous media. *Computers and Geotechnics*, 25(4):227–246, 1999.

Müller, M., Charypar, D., and Gross, M. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003.

Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151. 2004a.

Müller, M., Schirm, S., Teschner, M., Heidelberger, B., and Gross, M. H. Interaction of fluids with deformable solids. *Journal of Visualization and Computer Animation*, 15(3-4):159–171, 2004b.

Müller, M., Solenthaler, B., Keiser, R., and Gross, M. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 237–244. ACM Press, New York, NY, USA, 2005.

Nitao, J. J. and Bear, J. Potentials and their role in transport in porous media. *Water Resources Research*, 32:225–250, 1996.

NVIDIA. CUDA: Compute Unified Device Architecture. `http://www.nvidia.com/cuda`, 2006.

O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. Graphical modeling and animation of ductile fracture. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 291–294. ACM, New York, NY, USA, 2002.

O'Brien, J. F. and Hodgins, J. K. Graphical modeling and animation of brittle fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

Oh, S., Kim, Y., and Roh, B.-S. Impulse-based rigid body interaction in SPH. *Comput. Animat. Virtual Worlds*, 20(2-3):215–224, 2009.

Onoue, K. and Nishita, T. Virtual sandbox. In *Proc. of Pacific Conference on Computer Graphics and Applications*, page 253. 2003.

Pabst, S., Thomaszewski, B., and Straßer, W. Anisotropic friction for deformable surfaces and solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA 09*, page 149. 2009.

Pauly, M., Keiser, R., Adams, B., Dutré, P., Gross, M., and Guibas, L. J. Meshless animation of fracturing solids. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 957–964. ACM Press, New York, NY, USA, 2005.

Pla-Castells, M., García-Fernandez, I., and Martinez-Dura, R. J. Physically-based interactive sand simulation. In Mania, K. and Reinhard, E., editors, *Eurographics 2008 - Short Papers*, pages 21–24. 2008.

Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. T. Particle-based simulation of fluids. *Eurographics 2003 / Computer Graphics Forum*, 22(3):401–410, 2003.

Reeves, W. T. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983.

Robinson-Mosher, A., English, R. E., and Fedkiw, R. Accurate tangential velocities for solid fluid coupling. In *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 227–236. ACM, New York, NY, USA, 2009.

Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., and Fedkiw, R. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27(3):1–9, 2008.

Rungjiratananon, W., Szego, Z., Kanamori, Y., and Nishita, T. Real-time animation of sand-water interaction. In *Computer Graphics Forum (Pacific Graphics 2008)*, volume 27, pages 1887–1893. 2008.

Sawley, M., Cleary, P., and Ha, J. Modelling of flow in porous media and resin transfer moulding using smoothed particle hydrodynamics. In *Second International Conference on CFD in the Minerals and Process Industries*, pages 473–478. 1999.

Scheidegger, A. E. *The Physics of Flow through Porous Media*. University of Toronto Press and Oxford University Press, 1957.

Sin, F., Bargteil, A. W., and Hodgins, J. K. A point-based method for animating incompressible flow. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2009.

Solenthaler, B. and Pajarola, R. Density contrast sph interfaces. In *Proceedings of ACM SIGGRAPH / EG Symposium on Computer Animation*, pages 211–218. 2008.

Solenthaler, B. and Pajarola, R. Predictive-corrective incompressible sph. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–6. ACM, New York, NY, USA, 2009.

Solenthaler, B., Schläfli, J., and Pajarola, R. From fluid to solid and back in a unified particle model. In *Posters of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. 2006.

Solenthaler, B., Schläfli, J., and Pajarola, R. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

Stam, J. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.

Sumner, R., O'Brien, J. F., and Hodgins, J. K. Animating sand, mud, and snow. *Computer Graphics Forum*, 18(1):17–26, 1999.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. Elastically deformable models. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 87, pages 205–214. 1987.

Terzopoulos, D., Platt, J., and Fleischer, K. Heating and melting deformable models (from goop to glop). In *Proceedings of Graphics Interface*, pages 219–226. 1989.

Thürey, N., Keiser, R., Rüde, U., and Pauly, M. Detail-Preserving Fluid Control. In Cani, M. and Brien, J. O., editors, *Proceedings of the 2006 Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 7–15. ACM, 2006.

Tonnesen, D. Modeling liquids and solids using thermal particles. In *Proceedings of Graphics Interface*, pages 255–262. 1991.

Twomey, S. A., Bohren, C. F., and Mergenthaler, J. L. Reflectance and albedo differences between wet and dry surfaces. *Applied Optics*, 25:431–437, 1986.

Vignjevic, R., Campbell, J., and Libersky, L. A treatment of zero energy modes in the smoothed particle hydrodynamics method. *Godunov Methods: Theory and Applications*, 1995.

Wang, H., Mucha, P. J., and Turk, G. Water drops on surfaces. *ACM Trans. Graph.*, 24(3):921–929, 2005.

Wojtan, C., Carlson, M., Mucha, P. J., and Turk, G. Animating corrosion and erosion. In *Eurographics Workshop on Natural Phenomena*, pages 15–22. 2007.

Wojtan, C., Thürey, N., Gross, M., and Turk, G. Deforming meshes that split and merge. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 2009.

Yan, H., Wang, Z., He, J., Chen, X., Wang, C., and Peng, Q. Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds*, 20(2-3):417–426, 2009.

Zhu, Y. and Bridson, R. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 965–972. ACM Press, New York, NY, USA, 2005.

Zhu, Y. and Fox, P. J. Smoothed particle hydrodynamics model for diffusion through porous media. *Transport in Porous Media*, 43(3):441–471, 2001.

Zhu, Y., Fox, P. J., and Morris, J. A pore-scale numerical model for flow through porous media. *International journal for numerical and analytical methods in geomechanics*, 23(9):881–904, 1999.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Computer Science
Research Group Computer Graphics
Celestijnenlaan 200 A box 2402
B-3001 Leuven