University of Alberta

Physical-Based Non-Newtonian Fluid Animation Using SPH

By

Hai Mao ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**

Department of Computing Science

Edmonton, Alberta

Fall, 2006

Your file Votre référence
ISBN: 978-0-494-23078-7
Our file Notre référence
ISBN: 978-0-494-23078-7

Abstract

Fluids are commonly seen in our daily lives. They exhibit a wide range of motions, which depend on their physical properties, and often result in amazing visual phenomena. Hence, fluid animation is a popular topic in computer graphics. The animation results not only enrich a computer-generated virtual world but have found applications in generating special effects in motion pictures and in computer games.

The three-dimensional (3D) Navier-Stokes (NS) equation is a comprehensive mechanical description of the fluid motions. Smoothed Particle Hydrodynamics (SPH) is a popular particle-based fluid modeling formulation. In physical-based fluid animation, the fluid models are based on the 3D NS equation, which can be solved using SPH based methods.

Non-Newtonian fluids form a rich class of fluids. Their physical behavior exhibits a strong and complex stress-strain relationship which falls outside the modeling range of Newtonian fluid mechanics. In physical-based fluid animation, most of the fluid models are based on Newtonian fluids, and hence they cannot realistically animate non-Newtonian fluid motions such as stretching, bending, and bouncing.

Based on the 3D NS equation and SPH, three original contributions are presented in this dissertation, which address the following three aspects of fluid animation: (1) particle-based non-Newtonian fluids, (2) immiscible fluid-fluid collision, and (3) heating non-Newtonian fluids. Consequently, more varieties of non-Newtonian fluid motions can be animated, which include stretching, bending, and bouncing.

Acknowledgments

First of all, I would like to deeply thank my supervisor, Dr. Yee-Hong Yang. He kindly took me as his student when I was wandering in my research a few years ago. Also, it was his advice that guided me towards the research of non-Newtonian fluid animation, which is an exciting and rewarding topic. Over these years, Dr. Yang has been considerably and constantly offering me the advice not only on how to conduct meaningful research but also on how to become a decent scholar. Without his understanding, patience, support, guidance, and encouragement, I would not be able to finish this dissertation.

I would also like to thank the members of my examining committee: Dr. Walter Bischof, Dr. Pierre Boulanger, Dr. Yau Shu Wong, and Dr. Dimitris Metaxas for reading and commenting on this dissertation during their precious time. Dr. Sherif Ghali was my program committee member and also provided valuable comments to this dissertation.

I am also grateful to my previous supervisor, Dr. Mark Green. With his acceptance and support, I became a PhD student and had a smooth start of study at the University of Alberta.

In addition, I would like to thank the current and former members of the computer graphics research group at the Univercity of Alberta, in particular, Minglun Gong, Xuejie Qin, Daniel Neilson, Jiayuan Zhu, Yi Xu, Tong Guan, Nathan Funk, Cheng Lei, Danielle Sauer, Jason Selzer. Over the years, the inspiring discussions I had with them had important impact to this dissertation.

Finally, the financial support from NSERC, and the financial and administrative supports from the Department of Computing Science and the University of Alberta are gratefully appreciated.

To my parents for their continuous support

throughout the long long journey towards this degree.

To my wife who took me as her significant other when

this study could end without success,

and now, with whom this degree is more enjoyable.

Table of Contents

List of Tables

List of Figures

List of Equations

Chapter 1. Introduction

1.1 Physical-Based Fluid Animation

Fluids are commonly seen in our daily lives and they appear in many different forms such as a running river and a stormy ocean, in mixing coffee with cream, and a beer with foams. Due to their ever-changing shapes, fluids exhibit a wide range of motions, which depend on their physical behaviors, and often result in stunning visual phenomena, which makes fluid animation an attractive research topic in computer graphics (CG). The animation results enrich a computer-generated virtual world and have been used in many application areas including advertising, education, entertainment, flight simulation, scientific simulation, television, and so on.

In some literature, fluids include both liquids and gases because they both can be described with the Navier-Stokes equation. A major difference between them is that liquids are incompressible and gases compressible. The incompressibility is enforced with the continuity equation. Meanwhile, in a large amount of literature, fluids are only referred to as incompressible liquids. This terminology is followed in this dissertation.

Physical-based fluid animation means animating fluids based on the physical fluid models in Computational Fluid Dynamics (CFD). Many non-physical-based fluid models attempt to animate fluids [HNC02; OFL01; TKY02] or animate soft-bodies with fluid behaviors [DC96; DC98; TPF89; WMW86]. These models are not based on any physical fluid models. Compared with them, the physical-based fluid models can animate complex fluid phenomena more easily. This is because the parameters that control fluid behaviors have physical significance and thus are more intuitive to an animator. The discussion of non-physical-based fluid models is beyond the scope of this dissertation.

According to the current research trend, there are some common requirements for physical-based fluid animation. First, the graphic effects of the animation are the most important and are what the application users expect from the animation system. The second is that the animation should be produced within a reasonable amount of time. It is impractical that the time is counted by days or even weeks. Third, a complex fluid phenomenon usually consists of many types of fluid motions, all of which should be realistically animated in order to generate visually interesting results. For example, pouring water into a tank usually generates spraying, splashing, and waving. The spraying occurs when the water hits the bottom of the empty tank, the splashing occurs when the tank is partially filled with water, and the waving occurs before and after the pouring stops. This dissertation is concerned with the above mentioned requirements. Other requirements may include ease of use of the fluid models, compatibility with other animation systems, and so on.

One general rule to evaluate an animation method is based on the realism of results it produces. The more realistic, the better is the method. However, realism is very difficult to measure quantitatively and is often subjectively judged by human observers. For a simple setup, an animation could be placed side by side with the real video of the same phenomenon being animated, and then the quality of the animation could be evaluated by a human observer. However, as the complexity of the animated fluid phenomena increases, it is extremely difficult, if not impossible, and perhaps expensive to create a physical mockup that can produce the same phenomena. Furthermore, some of the parameters that control the animation may not have corresponding realization in nature. Hence, in the research, fluid animations are usually evaluated by human observers with common knowledge of fluid motions. For example, it is common knowledge that when a hard ball drops into a water tank, water droplets are splashing and thin sheets of water are being formed. This subjective evaluation approach is used in the recent fluid models [FF01; EMF02; MCG03; PTB*03; GBoB04; CMT04; MSKG05; HK05; LIGF06].

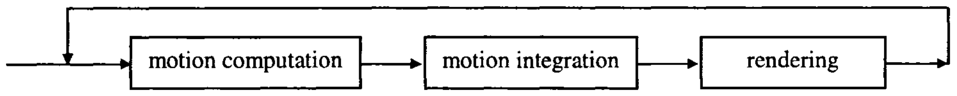


Figure 1-1: Three basic steps in a fluid animation loop.

A physical-based fluid animation is often produced with a loop procedure. The detailed operations in the loop may differ depending on the fluid models. But, the loop typically consists of three basic steps: (1) compute the fluid motion based on a dynamical model, (2) integrate the fluid motion over one time step, and (3) render the fluid into an image. The loop and the three basic steps are illustrated in Figure 1-1. Usually, the *n*th execution of the loop is labeled with time step *n*. With the continuous executions of the loop, the sequence of the rendered images results in an animation if they are displayed at an appropriate speed, e.g. 30 frames per second.

The three basic steps in the animation production loop correspond to the three issues in physical-based fluid animation, namely: (1) fluid motion computation, (2) fluid motion integration, and (3) fluid rendering. The fluid motion computation has drawn the attentions in both the CFD and CG communities. It also more or less determines how the other two issues are dealt with in a fluid model. This dissertation focuses on the fluid motion computation. The methods for fluid motion integration include: the Euler method, the leap-frog method, the prediction-relaxation method, the predictor-corrector method, the penalty method, and so on [Sta99; MCG03; PTB*03; MSKG05; CBP05; MY06]. For a specific fluid model, an appropriate integration method is required. The integration time step may be constant or variable depending on the motion computation. Fluid rendering depends on the fluid surface, which is a boundary between the fluid and air or between two or more fluids. It is not practical to directly render the volume of a fluid because the volume data is too much to render within a reasonable amount of time. The rendering methods usually simulate the fluid volume effects, such as transparency and refraction, based on the inside/outside normals of the fluid surface [Kaj86]. A fluid surface is represented by either explicit or implicit surfaces [MP89; KM90; OH95; FM96; FF01; EMF02; MCG03; PTB*03]. An explicit surface is typically a triangular mesh, which can be efficiently rendered by taking advantage of computer graphics hardware. However, for a fluid that involves large changes in shape and topology, it is hard to maintain the quality and compatibility of the triangular mesh. To overcome this difficulty, implicit surfaces are widely used to represent complex fluid surfaces [MCG03; MST*04; MY05; MSKG05].

Fluid motions are extensively studied in CFD. The physical fluid models are specially designed for accurate computation of fluid motions. And the computational results are expected to be consistent with the data obtained from physical experiments. Under this requirement, a fluid simulation normally runs for days or even weeks because, in order to obtain accurate results, the fluid is modeled at a very fine resolution and the motion is integrated at very small time steps. In computer graphics, however, it is impractical or unnecessary to produce fluid animations in such detail as required in CFD. Physical-based fluid animation is concerned only with the appearance more than the physical accuracy. The inaccuracy is acceptable as long as the animation looks realistic or interesting. In fact, in order to produce dramatic special effects in some cases, it is required to generate animations that are not physically realizable because physically realistic animations may not be visually appealing. Thus, the physical fluid models usually have to be customized for use in computer graphics. The physical accuracy and correctness must be traded off for less computational time as well as visual appeal. This is the general principle in dealing with fluid motion computation.

The three-dimensional (3D) Navier-Stokes (NS) equation is a well-known mechanical description of the fluid motions, and is the foundation of many advanced CFD research works. The earliest work of physical-based fluid animation appeared more than a decade ago [MP89; KM90]. At that time, the 3D NS equation was computationally too expensive to solve in practice due to limited computer resources [CL95]. Along with the advancement of computer hardware and the development of more efficient solution methods [Sta99; MCG03], the 3D NS equation is now widely used in physical-based fluid models and hence is used in this dissertation.

According to the ways of solving the 3D NS equation, the physical-based fluid models can be divided into two groups: the grid-based models [KM90; CL95; FM96; Sta99; FF01; EMF02; LP02; HK03; TFK*03; CMT04] and the particle-based models [Roy95; MCG03; PTB*03; MST*04; MSKG05; MY06]. It should be pointed out that the two types of the models are physically equivalent and they model fluids just from two different points of view.

In a grid-based model, a fixed grid is created in the animation space which is or may be occupied by the animated fluid. The fluid attribute values are attached to the grid. The fluid motion is computed according to the fluid values on the grid. Since the moving fluid surface does not fit well with the grid which is aligned with three orthogonal axes of the Cartesian coordinate system, markers (or massless particles) are distributed within the grid to mark the fluid surface and are driven by the velocity field attached to the grid.

In a particle-based model, a fluid volume is represented by an aggregation of particles. The fluid attribute values such as velocities and pressures are associated with the particles. The fluid motion of a particle is computed according to the fluid attribute values. The particles each have mass and are driven by the fluid dynamics. The fluid surface motion is characterized by the motions of surface particles.

A particle-based model has the following advantages over a grid-based model:

(1) The animation space is not limited to the pre-generated grid, which is a prerequisite for all grid-based models. Because of the grid, even with markers, it is still difficult, if not impossible, for the grid-based models to deal with free fluid surfaces, interfaces, and deformable boundaries in the animation space, all of which can be easily handled with the particle-based models [LL03].

(2) The fluid motions are governed by ordinary differential equations, which do not have the advection term and are easier to solve than the partial differential equations used in grid-based models.

(3) Fluid motions such as waving, splashing, and spraying are well handled within the same particle-based framework. No special attention is needed for each type of motion;

whereas, special markers are required in grid-based models in order to trace large deformed surfaces such as in splashing and spraying [FM96; Sta99; FF01; EMF02].

(4) It is straightforward to model the interactions of several fluid phases bounded by free surfaces [PTB*03].

Smooth Particle Hydrodynamics (SPH) is a popular particle-based fluid formulation and has been applied to address many fluid motion issues in CFD [Mon92; LL03; Mon05]. In computer graphics, SPH has been widely used in many physical-based graphical models [Roy95; SF95; DC96; DC99; SAC*99; MCG03; MST*04; CBP05]. Among them, Roy [Roy95] uses SPH to animate weakly compressible fluid, and Muller et al. [MCG03] use SPH for interactive fluid animation. These successes show that SPH is a very flexible formulation for a variety of fluid motions. Another particle-based fluid formulation developed in CFD is called Moving Particle Semi-implicit (MPS). It is used in a physical-based fluid model by Premoze et al. [PTB*03] to animate fluids. Compared with their MPS-based model, a SPH-based particle model is more flexible with the particle mass, the interaction radius, and the weighting function. Thus, SPH is employed in this dissertation and the corresponding animation framework is described in Chapter 3.

So far, many physical-based fluid models have been proposed. They are able to animate various fluid motions. For example, photo-realistic fluid animations are produced with the grid-based models by Foster et al. [FF01] and Enright et al. [EMF02]. In their example animations, the water surface waving and splashing are impressively presented. However, due to the complexity of the fluid behaviors, it is still very difficult to realistically animate many fluid phenomena, especially non-Newtonian fluid phenomena.

1.2 Non-Newtonian Fluids

Traditional Newtonian fluid motions are governed by the interplay of viscous and inertial forces. If free surfaces are involved, surface tension is also an important factor. The 3D NS equation accommodates all these factors and is an elegant and comprehensive dynamics description for Newtonian fluids. However, in our daily lives, there exist many examples of non-Newtonian fluids, such as egg white and blood, whose motions fall outside the scope of Newtonian fluid mechanics.

In fluid dynamics, fluids are classified as Newtonian and non-Newtonian fluids. A fluid is non-Newtonian if the stress tensor cannot be expressed as a linear, isotropic function of the velocity gradient; otherwise, it is Newtonian [OP02]. The stress tensor is a 3x3 matrix, and expresses the internal stress which a fluid develops in response to being deformed. The fluid deformation is described with the velocity gradient. In essence, the molecules of a Newtonian fluid are too small for their dynamics, i.e. the internal stress, to influence substantially the macroscopic fluid behaviors. A perfect example of Newtonian fluids is water. The value of its stress tensor is too small to react to its deformation. In contrast, non-Newtonian fluids have complex microstructures at much larger scales than the atomic scale. The interaction between the fluid deformation and the stress exerted by such microstructures is too significant to be ignored.

The constitutive equation is a mathematical description of how the stress is determined by the deformation. It is also called the constitutive law in some CFD literature [Ren00]. Due to the complexity of non-Newtonian microstructures, the constitutive equation is usually a differential equation:

$$\frac{dT}{dt} = F(T, \mathbf{v})$$

Equation 1-1: Generic form of a differential constitutive equation.

where t is the time, T the fluid stress tensor, and $\mathbf{v}$ the fluid velocity at any location in the fluid at t. Equation 1-1 computes the stress tensor rate as a function of the fluid stress and the fluid velocity. Then, the stress tensor can be obtained from integrating the stress tensor rate. Also due to the variety of non-Newtonian microstructures, there is no unified constitutive equation for all non-Newtonian fluids. Many different constitutive equations were proposed over the years [Ren00; OP02]. Among them, the Maxwell model and its variants are very popular, and are adopted in computer graphics as well as in this dissertation.

In addition to differential constitutive equations, the fluid stress can also be determined with integral constitutive equations. However, differential constitutive equations are more thoroughly investigated than integral constitutive equations, since they are easier to implement for numerical simulation [Ren00]. For brevity, differential constitutive equations are referred to as constitutive equations in the rest of this dissertation. More information about non-Newtonian fluids can be found in CFD literature [Ren00]. Also, non-Newtonian fluids are studied under a major scientific discipline known as Rheology, which is the study of the deformation and flow of matter under the influence of an applied stress [OP02].

The constitutive equation is a tensor-based fluid stress model. In computer graphics, the fluid stress can also be modeled by a linear combination of elastic spring, viscous dashpot, and the von Mises yield condition [TF88; CBP05]. This combination has been used to model linear viscoelastic fluids, which are examples of non-Newtonian fluids [OP02].

Newtonian and non-Newtonian fluids share many fluid properties such as surface tension and incompressibility. Many animation methods for Newtonian fluids are also applicable to non-Newtonian fluids. According to the fluid dynamics classification, the major difference between them is how the fluid stress is related to the fluid strain, which is described by the constitutive equation. Simply speaking, the less fluid stress is incurred by the fluid strain, the more a fluid behaves like a Newtonian fluid. In this sense, Newtonian fluids can be treated as special cases of non-Newtonian fluids. It is the stress-strain relationship that cannot be handled properly by the Newtonian fluid models and must be accommodated in order to realistically animate non-Newtonian fluids. In terms of the visual effects, Newtonian fluids are characterized with the inelastic motions such as waving, flowing, splashing, and spraying. Meanwhile, non-Newtonian fluids exhibit the elastic motions such as stretching, bending, and bouncing, which cannot be animated by the Newtonian fluid models but can be by the fluid models described in this dissertation.

So far in computer graphics, a few attempts [TF88; GBoB04; CBP05] have been made to animate non-Newtonian fluids. According to their demonstrations, their models can realistically animate only a small fraction of the whole non-Newtonian fluid behaviors. More efficient and comprehensive techniques are desired in generating the myriads of non-Newtonian fluid phenomena.

1.3 Contributions

In physical-based fluid animation, the majority of fluid models focus on Newtonian fluids. Meanwhile, non-Newtonian fluids exhibit a rich variety of amazing phenomena, and their dynamics is more complex than that of Newtonian fluids. In order to realistically animate non-Newtonian fluids, the animation techniques for Newtonian fluids are insufficient, and more powerful fluid models than the existing ones are required. This dissertation focuses on the physical-based non-Newtonian fluid animation. It has three original contributions that correspond to three new fluid models.

The first contribution is a new particle-based non-Newtonian fluid model. To simulate non-Newtonian fluid stress in computer graphics, previous grid-based non-Newtonian fluid models use a quasi-linear Maxwell model and the von Mises yield condition, while previous particle-based non-Newtonian fluid models use a linear combination of elastic spring, viscous dashpot, and the von Mises yield condition. In comparison, the new model employs a non-linear Maxwell model and is more accurate in describing non-Newtonian fluid motions involving rotations. In addition, it is observed that fluid deformations may cause poor particle distribution which in turn causes inaccurate fluid modeling. To address this problem, the new model includes a new particle re-sampling method, in which particles are down-sampled and then up-sampled such that a good distribution of particles is attained.

The second contribution is a new particle-based immiscible fluid-fluid collision model. In previous physical-based fluid models, fluid interactions with rigid and deformable solids are simulated, so is the interface tension between fluids. In contrast, the new model simulates the impulsive collisions between fluids with a simple particle collision method. A useful modeling feature is that the new model not only can prevent immiscible fluids from mixing with each other, but also can allow one fluid to run through or to wrap around another fluid. The new model is very flexible and can work with many existing particle-based fluid models.

The third contribution is a new particle-based heating model for non-Newtonian fluids. The heat equation in previous heating models for fluids is based on an assumption that the fluid density is constant. In the new heating model, a more flexible heat equation is employed which accommodates variable fluid density. Thus, the heat transfer simulation is more physically reliable. In addition, the new heating model simulates the heat exchange between immiscible fluids, and controls the non-Newtonian fluid property with variable fluid temperature. As a result, different elastic behaviors including solid-like and fluid-like behaviors can be animated in different parts of a fluid body.

In order to demonstrate the contributions, the three new fluid models are implemented and the example fluid animations are produced and presented in this dissertation. Overall, the animated fluids can drip, flow, split, merge, crouch, stretch, bend, and bounce. They can exchange heat and interact with each other. These behaviors fall between those of solids and Newtonian fluids.

1.4 Dissertation Organization

In Chapter 2, the previous work in the physical-based fluid animation is reviewed. In Chapter 3, the basic framework for physical-based fluid animation using SPH is described. In the following three chapters, 4, 5, and 6, the three contributions are presented, respectively: (1) particle-based non-Newtonian fluids, (2) immiscible fluid-fluid collision, and (3) heating non-Newtonian fluids. Finally, the conclusion and future work are given in Chapter 7.

Chapter 2. Previous Work

In this chapter, previous physical-based fluid models are reviewed. Because their underlying physical fluid models are based on the Navier-Stokes (NS) equation or its variants, they are divided into three groups: the height field models, the grid-based models, and the particle-based models. The height field models are based on the shallow water equations, which can be simplified from the NS equation. The grid-based models are based on the Eulerian version of the NS equation, and the particle-based models on the Lagrangian version of the NS equation.

2.1 Height Field Models

The height field models are based on the following shallow water equations:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + g\frac{\partial H}{\partial x} = 0$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + g\frac{\partial H}{\partial y} = 0$$

$$\frac{\partial H}{\partial t} + \frac{\partial}{\partial x}(uH) + \frac{\partial}{\partial y}(vH) = 0$$

Equation 2-1: Shallow water equations.

where *u* and *v* are the water velocity components along the *x*- and *y*- axis, respectively, *t* denotes the time, *g* the gravitational constant, and *H* the water surface height. The shallow water equations are derived from the NS equation by assuming zero viscosity and considering only two dimensional motions. The first two equations are derived from Newton's second law of motion, and the third one is the continuity equation for volume conservation. In modeling water motion, the shallow water equations require three assumptions. The first one is that the water surface can be represented by a height field. This assumption has two obvious limitations: the water cannot splash and water waves cannot break. However, as long as the forces on the water are sufficiently gentle, this assumption will not introduce significant errors. The second assumption is that the vertical velocity component of the water can be ignored. Once again, the limitation is fairly clear. If a disturbance creates very steep waves on the water surface, the equations will not be accurate. The third assumption is that the horizontal velocity component of the water in a vertical column is approximately constant. If there is turbulent flow or unusually high friction in the bottom, this assumption will not hold. Nonetheless, the shallow water equations adequately model a reasonably large range of water surface motions. They are adopted in the height field models for simplicity and for short motion computational time.

Kass and Miller [KM90] propose a solution to the shallow water equations. They begin with the shallow water equations for a height-field surface in 2D:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + g\frac{\partial H}{\partial x} = 0$$

$$\frac{\partial H}{\partial t} + \frac{\partial}{\partial x}(uH) = 0$$

Equation 2-2: Shallow water equations in 2D.

With two more assumptions that the water velocity is small and the depth is slowly varying, the equations become:

$$\frac{\partial u}{\partial t} + g\frac{\partial H}{\partial x} = 0$$

$$\frac{\partial H}{\partial t} + H\frac{\partial u}{\partial x} = 0$$

Equation 2-3: Simplified shallow water equations in 2D.

Using appropriate differentiations and substitutions, Equation 2-3 becomes:

$$\frac{\partial^2 H}{\partial t^2} = gH\frac{\partial^2 H}{\partial x^2}$$

Equation 2-4: Merged shallow water equation in 2D.

With the extension to 3D, Equation 2-4 becomes:

$$\frac{\partial^2 H}{\partial t^2} = gH\left(\frac{\partial^2 H}{\partial x^2} + \frac{\partial^2 H}{\partial y^2}\right) = gH\nabla^2 H$$

Equation 2-5: Merged shallow water equations in 3D.

To obtain numerical solutions to both Equation 2-4 and Equation 2-5 is very simple and easy to program. With this model, water surface motions such as waves do not need to be explicitly specified because they arise naturally from the physical conditions occurring within the animation system. The model can be used for a large volume of fluid without incurring a huge computational cost because the computational time is linear with respect to the number of samples of the height field.

O'Brien and Hodgins [OH95] extend the above height field model, and propose three fluid sub-models: the volume model, the surface model, and the spray model. In the volume model, the main fluid body is divided into a rectilinear grid of connected columns. The flow between these columns occurs through a set of virtual pipes that connect adjacent columns. A typical column has eight virtual pipes: four of them connect to the four axis-aligned neighboring columns and the other four connect to the four diagonal neighboring columns. The flow in these virtual pipes is determined by the difference in hydrostatic pressure between the columns that they connect. The static pressure P_{ij} of a column in the grid at position $[i, j]$ is:

$$P_{ij} = H_{ij}\rho g + P_0 + E_{ij}$$

Equation 2-6: Fluid column pressure.

where ρ is the fluid density, g the gravitational acceleration, P_0 the atmospheric pressure, and E_{ij} the external pressure. H_{ij} is the height of the column at $[i, j]$ and is obtained by:

$$H_{ij} = \frac{V_{ij}}{r_x r_y}$$

Equation 2-7: Fluid column height.

where V_{ij} is the volume of that column, and r_x and r_y the nominal distance between the defined grid points in the x and y directions, respectively. The difference in pressure between two adjacent columns causes fluid flowing in the virtual pipe connecting them. The volume of each column is updated by the fluid flowing in or out of the virtual pipes at each time step. If a column has a negative volume, an adjustment procedure is performed until the negative volume is eliminated. Incompressibility is enforced by the conservation of flow.

In the surface model, a rectilinear grid of control points defines the surface mesh. The control points are mapped onto the column grid of the volume model such that each control point is surrounded by the four columns. The vertical position z_{ij} of a control point is computed by the average of the heights of the four columns surrounding that control point:

$$z_{ij} = \frac{H_{i,j} + H_{i,j+1} + H_{i+1,j} + H_{i+1,j+1}}{4}$$

Equation 2-8: Vertical position of control point on fluid surface.

The force $\mathbf{f}_e$ applied to a control point by external objects is reformulated as an external pressure E_{ij}:

$$E_{ij} = -\frac{\mathrm{f}_e}{4r_x r_y}$$

Equation 2-9: External pressure on control point.

The negative sign is due to the downward force that results in an increase in external pressure. The external pressure is evenly distributed on the four columns surrounding that control point.

In the spray model, particles are employed to model droplets that are disconnected from the main fluid body. When an area of the surface has an upward velocity greater than a user-specified threshold, particles are distributed uniformly over that area and the initial velocities for the particles are interpolated from the surface velocities. The surface velocities are determined by the column's vertical velocity and the flow velocities in the virtual pipes. Once created, the particles fall under gravity and do not interact with each other. Using this spray model, the generated droplets are uniform and the animated spraying shows symmetric patterns. This behavior is not realistic in comparison with the randomness as appeared in real fluid spraying.

When an object collides with a fluid surface, the motions of the fluid and of the object are computed from the collision forces that are equal in magnitude but opposite in sign. The force $\mathbf{f}_e$ on the fluid surface is used to compute the external pressure E_{ij}. The force on the object is used to compute the object position. The object has to be floating on the surface. It cannot bounce off or submerge into the fluid.

The model by O'Brien and Hodgins has the advantages of the previous height field model [KM90], and animates a wider range of fluid behaviors such as splashes caused by impacts and the interactions of floating objects with the fluid.

Recently, Layton and van de Panne [LP02] propose to solve the complete shallow water equations using an implicit semi-Lagrangian integration scheme. They consider the shallow water equations in 2D in Lagrangian form:

$$\frac{du}{dt} + g\frac{\partial H}{\partial x} = 0$$

$$\frac{dH}{dt} + H\frac{\partial u}{\partial x} = 0$$

Equation 2-10: Shallow water equations in 2D in Lagrangian form.

where the Lagrangian derivative is defined as:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x}$$

Equation 2-11: Lagrangian derivative in 1D.

Then, the solutions to the 2D and 3D shallow water equations are presented. The solution equations are quite lengthy and not given here for brevity. The proposed model produces the example animations in real-time, and the authors claim that the model can handle floating objects although no example animation is given in the paper. The model is compared in computational time with the early height field model [KM90], but not with the more advanced height field model [OH95]. Nonetheless, the proposed model is based on the complete shallow water equations with fewer simplifying assumptions than the previous models [KM90, OH95], and is capable of producing realistic water waves. Because of the semi-Lagrangian integration scheme, it is also stable even with large time steps.

In general, the height field models are suitable for interactive animations because of the simple underlying physical model. However, the three assumptions described at the beginning of this section make the height field models incapable of modeling 3D flows, fluids with high viscosity, and breaking waves. Fluid splashing and spraying were attempted in [OH95]. But differences between the images of the real motions and the animated motions are quite obvious.

2.2 Grid-Based Models

The Navier-Stokes (NS) equation is a comprehensive fluid motion description at any location in the fluid at any instant of time. It is the momentum equation derived from Newton's second law of motion. The Eulerian version of the NS equation is written as:

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\frac{1}{\rho}\nabla P + \mu \nabla^2 \mathbf{v} + \mathbf{f}$$

Equation 2-12: Eulerian version of the NS equation.

where $\mathbf{v}$ is the fluid velocity, t the time, ρ the fluid density, P the fluid pressure, μ the fluid kinematic viscosity, and $\mathbf{f}$ the summation of external forces such as gravity. For brevity, the Eulerian version of the NS equation is referred to as the NS equation in this sub-section. The NS equation couples the velocity and pressure fields and is more complex than the shallow water equations. The existing solutions to the NS equation in CFD are computationally expensive for use in computer graphics. For the modeling of incompressible fluids, the NS equation is usually applied in conjunction with the continuity equation:

$$\nabla \cdot \mathbf{v} = 0$$

Equation 2-13: Continuity equation.

Since the NS equation is typically solved over a fixed grid, the corresponding models are grid-based models, which are discussed in this sub-section.

Chen and Lobo [CL95] propose a grid-based fluid model with the NS equation. To avoid the expensive solution to the 3D NS equation, they solve the 2D NS equation so that the computational complexity reduces from the cube of the resolution to the square of the resolution. Then, they raise the surface of the fluid according to the corresponding pressures in the 2D flow field. In particular, the NS equation without external forces is rewritten in dimensionless form:

$$\frac{\partial \mathbf{v}}{\partial t} + u\frac{\partial \mathbf{v}}{\partial x} + v\frac{\partial \mathbf{v}}{\partial y} + w\frac{\partial \mathbf{v}}{\partial z} + \nabla P = \frac{1}{Re}\nabla^2 \mathbf{v}$$

Equation 2-14: NS equation in dimensionless form.

where u, v and w are the three velocity components along the x-, y- and z- axis, respectively, and Re is the Reynolds number. The continuity equation is approximated with:

$$\varepsilon P + \nabla \cdot \mathbf{v} = 0, \quad \varepsilon > 0\,.$$

Equation 2-15: Approximated continuity equation.

The joint solution of Equation 2-14 and Equation 2-15 tends toward the solution of the NS equation and the continuity equation as $\varepsilon \longrightarrow 0$. Equation 2-14 and Equation 2-15 in 2D are solved using the staggered marker-and-cell discretization. After the calculations of the velocity vectors and pressures of the 2D fluid flow field, the surface height in the third dimension at grid (i, j) is computed from a scaled value of P_{ij}. Due to the simplification from 3D to 2D, the proposed model is able to animate the interaction between moving objects and the flow field at an interactive-rate. However, the proposed model loses the depth information, and the modeling of the moving objects and boundary conditions are restricted to two dimensions. The more complex fluid motions involving the depth information are beyond the capability of the proposed model.

In CG, Foster and Metaxas [FM96] are the first to propose a solution to the complete 3D NS equation. The computation domain is divided into a fixed coarse rectangular grid aligned with the Cartesian coordinate system. The coarse resolution is the trade off between efficiency and accuracy. A grid cell may be in one of the following states: 1) it contains a *solid* obstacle, 2) it is filled with *fluid*, 3) it is a *surface* cell on the boundary between the fluid and the surrounding medium, or 4) it is *empty*. In all the four cases, the velocity and pressure fields are defined everywhere in the fluid. This discretization leads to an explicit finite difference solver to the NS equation. At the beginning of each iteration step, the boundary conditions are checked and set according to the locations of stationary obstacles, the inflow and outflow, and the topology of the free surface. In each iteration step, the velocity and pressure fields are integrated forward by the finite difference approximation, and then adjusted by a repetitive procedure until all the cells in the flow field have a divergence less than a small prescribed threshold. Once convergence is achieved, the fluid is considered to be locally incompressible and the next iteration step starts. The converged velocity and pressure fields can be used to compute the motion of buoyant objects.

Based on the solved fluid motion over a finite mesh, the position of the fluid surface is represented using three methods: marker particles, free surface particles, and height field. Marker particles are ideal for animating violent phenomena such as overturning waves because they define the position of the fluid exactly, regardless of how complex the surface has become. Free surface particles are placed along the boundaries between the fluid and the obstacles or air. They can be removed or inserted such that the surface always remains continuous and the colliding surfaces are smoothly connected. For fluid motion without overturning waves, the height field is used to represent the fluid surface. This height field is essentially different from the previous height field models. Here, the surface elevation is driven by the underlying fluid velocity. Therefore, velocity or pressure disturbances anywhere in the fluid volume can affect the surface.

The motions of rigid dynamic objects can be animated using the velocities and pressures calculated all over the fluid flow field. The floating objects must be small compared to the computational grid size such that they do not affect the flow. They act like large marker particles moving and rotating according to the local pressure field. Large objects that are able to influence the motion of the fluid are not modeled.

By solving the NS equation in three dimensions, the proposed model is capable of modeling a wide range of fluid behaviors that cannot be modeled using its previous fluid models in computer graphics. However, the main problem remains that the explicit numerical integration can become unstable with large time steps. Using small time steps will alleviate this problem but with increasing computational cost of animation.

Stam [Sta99] proposes a stable algorithm that solves the 3D NS equation on a grid. The algorithm is based on the mathematical result that any vector field **w** can uniquely be decomposed into a divergence-free vector field plus a gradient field:

$$\mathbf{w} = \mathbf{u} + \nabla q$$

Equation 2-16: Decomposition of vector field w.

where **u** is the divergence-free vector field, $\nabla \cdot \mathbf{u} = 0$, and q the scalar field. With this decomposition, an operator **P** can be defined that projects any vector field **w** onto a divergence-free vector field **u** = **Pw**. In the projection, a Poisson equation is solved first for the scalar field q:

$$\nabla \cdot \mathbf{w} = \nabla^2 q \,.$$

Equation 2-17: Poisson equation for scalar field.

Then, the divergence-free vector field **u** is computed as follows:

$$\mathbf{u} = \mathbf{P}\mathbf{w} = \mathbf{w} - \nabla q \,.$$

Equation 2-18: Computation of divergence-free vector field.

This projection operator is applied to both sides of the NS equation:

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{P}\left(-(\mathbf{v} \cdot \nabla)\mathbf{v} + \nu \nabla^2 \mathbf{v} + \mathbf{g}\right).$$

Equation 2-19: NS equation under projection operator.

Equation 2-19 is the fundamental equation from which the stable solver is developed.

The stable solver starts from an initial state, $\mathbf{w}_0(\mathbf{x}) = \mathbf{v}(\mathbf{x}, t)$, which is the velocity field from the previous time step at time t, where **x** denotes the fixed grid location. To compute the new velocity field $\mathbf{v}(\mathbf{x}, t+\Delta t)$ at time $t+\Delta t$, each term on the right hand side of

Equation 2-19 is sequentially solved, followed by projection onto the divergence-free field. This procedure consists of four steps: (1) force addition, (2) advection, (3) diffusion and (4) projection. In the first step, the external force **f**(**x**, t) is integrated and added:

$$\mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t\,\mathbf{f}(\mathbf{x},t)$$

Equation 2-20: Addition of external force.

The second step is to resolve the advection term by back-tracing to the position at time Δt ago:

$$\mathbf{w}_2(\mathbf{x}) = \mathbf{w}_1(f_b(\mathbf{x}, -\Delta t))$$

Equation 2-21: Computation of advection term with back-tracking.

where $f_b(\mathbf{x},-\Delta t)$ is the back-tracing function. The third step is to resolve the diffusion term using an implicit method:

$$(\mathbf{I} - \nu\Delta t\nabla^2)\mathbf{w}_3(\mathbf{x}) = \mathbf{w}_2(\mathbf{x})$$

Equation 2-22: Computation of diffusion term with implicit method.

where **I** is the identity operator. The fourth step is to project onto a divergence-free field by resolving the pressure term:

$$\nabla^2 P = \nabla\cdot\mathbf{w}_3(\mathbf{x}) \quad\Rightarrow\quad \mathbf{w}_4(\mathbf{x}) = \mathbf{w}_3(\mathbf{x}) - \nabla P$$

Equation 2-23: Computation of pressure term with projection operator.

The stable solver is claimed to be unconditionally stable at the cost of extensive dissipation. The example animations show 3D gaseous motions under real-time user interactions. However, the 3D fluid motions are not demonstrated. One limitation of the proposed model is that the stable solver must be supplemented with either *periodic* or *fixed* boundary conditions, which are not physically realizable and make it difficult to animate fluid of free surface motions.

Foster and Fedkiw [FF01] propose a hybrid fluid model for animating free surface motions of fluid. The fluid motion computation is based on the previous stable solver in [Sta99] and the incompressibility constraint is enforced using a similar Poisson equation as Equation 2-17. The proposed model animates viscous fluids ranging from water to thick mud. These fluids can mix freely and move arbitrarily within a fixed three-dimensional grid, and interact realistically with stationary or moving polygonal objects. The actual distribution of a fluid in the fixed three-dimensional grid is represented using an implicit function φ. The fluid surface is defined by the $\varphi=0$ isocontour with $\varphi\leq 0$ representing the fluid and $\varphi>0$ the air. To deal with arbitrary fluid surfaces, an implicit function is derived from a combination of inertialess particles and a dynamic level set. The level set provides a smooth surface to regions of fluid that are well resolved compared to the grid, whereas the particles provide details where the surface starts to splash. At each time step, the particles move forward according to the velocity field **v**:

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{v}(\mathbf{p}_i)$$

Equation 2-24: Particle movement by velocity field.

where $\mathbf{p}_i$ is the location of particle *i*. The level set value φ evolves over time also using the velocity field **v**:

$$\frac{\partial\varphi}{\partial t} + \mathbf{v}\cdot\nabla\varphi = \mathbf{0}$$

Equation 2-25: Level set evolution by velocity field.

For each particle near the surface, the surface curvature κ is calculated as:

$$\kappa = \nabla \cdot (\nabla \varphi / |\nabla \varphi|)$$

Equation 2-26: Surface curvature.

In regions that have low curvature, the particles are ignored and the level set function is used to give a smooth representation of the fluid surface. In regions of high curvatures, however, the particles are better indicators of the rough surface topology and are used to modify the local values of the level set function φ. Even with the tight coupling between the particles and the level set, some particles will escape the inside of the fluid layer since the grid is too coarse to represent them individually. These escaped particles indicate the presence of fluid spraying, and they are rendered directly as fluid droplets, in particular, uniform spheres. These small fluid drops have no interaction with each other. In reality, sprayed fluid drops are of different shapes and sizes and may split and merge with each other. As can be seen in the example animations, the rigid uniform small spheres complement the fluid surface motion but manifest the artifact in comparison with real fluid spraying. Therefore, the proposed model can handle fluid spraying but not yet realistically.

Enright et al. [EMF02] present a new water surface model for the animation and rendering of photorealistic water effects. In their model, a hybrid surface tracking method is proposed. The method is based on the work of Foster and Fedkiw [FF01], and also uses the inertialess particles combined with a dynamic implicit surface. In addition, a new treatment of the velocity at the surface is proposed to obtain a more visually realistic water surface. The motions of both the inertialess particles and the implicit function representing the surface are dependent on the velocities associated with the fixed underlying computational grid. By extrapolating velocities across the surface and into the region occupied by the air, more accurate and better visual results are obtained. The new techniques introduced in this model are justified to preserve fluid volume better [EFFM02]. It is not clear though if this model simulates fluid motions more accurately than the previous model [FF01]. Indeed, the resulting splash after the ball impacts the water surface is dramatically different between the two figures shown in the papers [EMF02] and [FF01]. Without comparing with the real events, it is difficult, if not impossible, to tell which one conforms more closely to the real fluid motion. It is also noted that, same as in the previous model [FF01], the proposed model does not address the issue of realistically fluid spraying animation.

More fluid phenomena are animated with the grid-based models than with the particle-based models. For examples, fluid melting and flowing are animated by Carlson et al. [CMvHT02], viscoelastic fluids by Goktekin et al. [GBoB04], bubbles in liquid by Hong and Kim [HK03; HK05], fluid with splash and foam by Takahashi et al. [TFK*03], and water drops on surfaces by Wang et al. [WMT05]. To animate fluid-solid interactions, Genevaux et al. [GHD03] propose an interface between the fluid and the solid, while Carlson et al. [CMT04] treat rigid solids as special fluids with rigid motions. Guendelman et al. [GSLF05] propose a coupling method for water to interact with deformable and rigid thin shells.

After all, the grid-based models have produced photorealistic animations of complex fluid motions. However, due to their nature of working on a grid, these models have difficulties to realistically animate fluids at small scales in comparison to the grid size. Thus, the animations with small details have to be produced with a fine grid, which is a time-consuming task.

2.3 Particle-Based Models

The particle system was introduced into computer graphics by Reeves [Ree83] more than two decades ago. Since then, it has become a popular tool for modeling various phenomena. Miller and Pearce [MP89] apply molecular dynamics for the particle interaction and animate limited behaviors of fluids and melting solids. Terzopoulos et al. [TPF89] use particles to model thermoelastic materials with various inter-particle forces. Tonnesen [Ton91] also uses molecular dynamics for modeling liquids and solids, and applies heat transfer to animate the material phase transition between liquids and solids.

In addition, many particle-based models have been proposed for fluid animation, and they are based on the Lagrangian version of the NS equation:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla P + \mu\nabla^2\mathbf{v} + \mathbf{f} ,$$

Equation 2-27: Lagrangian version of NS equation.

which is similar to the Eulerian version of the NS equation, Equation 2-12. Note that the expression $\partial\mathbf{v}/\partial t + (\mathbf{v}\cdot\nabla)\mathbf{v}$ on the left hand side of Equation 2-12 is replaced by the substantial derivative $d\mathbf{v}/dt$ in Equation 2-27. Since the particles move with the fluid, the substantial derivative of the velocity field is simply the time derivative of the velocities of the particles. As a result, the advection term $(\mathbf{v}\cdot\nabla)\mathbf{v}$ is absent. Equation 2-27 is an ordinary differential equation and is easier to solve than the corresponding Eulerian version of the NS equation, which is a partial differential equation. More detailed comparison between the Lagrangian and Eulerian versions of the NS equation can be found in [PTB*03; MCG03]. Once particle velocities are resolved using Equation 2-27, new particle positions are trivially integrated according to:

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}$$

Equation 2-28: Particle position integration from velocity.

where **p** is particle position.

Many particle-based fluid models are based on Smooth Particle Hydrodynamics (SPH). Roy [Roy95] proposes one of the early SPH-based models, in which the fluid incompressibility is simulated using a weakly compressible method. Desbrun and Cani utilize SPH to animate highly deformable bodies [DC96], and improve the efficiency with an adaptive scheme [DC99]. Muller et al. [MCG03] present a SPH-based model that can interactively animate fluid splashing and swirling. The SPH is also utilized to animate lava flows by Stora et al. [SAC*99], point-based elastic, plastic, and melting objects by Muller et al. [MKN*04], fluid-solid interaction by Muller et al. [MST*04], and fluid-fluid interactions by Muller et al. [MSKG05] and by Mao and Yang [MY06]. In Chapter 3, more detailed information about the SPH applications in computer graphics is given.

Other than the SPH, the Moving-Particle Semi-Implicit (MPS) is another particle-based fluid modeling formulation. Premoze et al. [PTB*03] propose a MPS-based fluid model that can produce appealing animations of fluid splashing and swirling. In the model, the particle interaction is weighted depending on the particle distance. The weight between two particles *i* and *j* is:

$$w(r) = \begin{cases} r_e/r & 0 \le r \le r_e \\ 0 & otherwise \end{cases}$$

Equation 2-29: Weight function in MPS.

where r_e is the interaction radius and $r = |\mathbf{p}_j - \mathbf{p}_i|$ is the distance between the two particles whose positions are $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively. Because uniform particles are used, the particle number density e_i for particle i is computed instead of the fluid density:

$$e_i = \sum_{j=1}^{n} w(|\mathbf{p}_j - \mathbf{p}_i|)$$

Equation 2-30: Particle number density in MPS.

where n is the number of neighboring particles to particle i. Any particles within the interaction radius to particle i are the neighboring particles. As the fluid is incompressible, the particle number density must be equal to a constant: e^0.

To solve the momentum Equation 2-27, differential operators on particles are defined. Let q and $\mathbf{u}$ be arbitrary scalar and vector quantities, respectively. The gradient of q_i on particle i is defined as:

$$\nabla q_i = \frac{D}{e^0} \sum_{j=1}^{n} \frac{q_j - q_i}{|\mathbf{p}_j - \mathbf{p}_i|^2} (\mathbf{p}_j - \mathbf{p}_i) w(|\mathbf{p}_j - \mathbf{p}_i|)$$

Equation 2-31: Scalar gradient in MPS.

where D is the number of dimensions and q_j the scalar quantity on particle j. Similarly, the gradient of $\mathbf{u}_i$ on particle i is:

$$\nabla \cdot \mathbf{u}_i = \frac{D}{e^0} \sum_{j=1}^{n} \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{p}_j - \mathbf{p}_i)}{|\mathbf{p}_j - \mathbf{p}_i|^2} w(|\mathbf{p}_j - \mathbf{p}_i|)$$

Equation 2-32: Vector gradient in MPS.

where $\mathbf{u}_j$ is the vector quantity on particle j.

The momentum Equation 2-27 is then solved by the semi-implicit method. After time step n, the external force and viscosity in the equation are computed explicitly. Temporary particle positions $\mathbf{p}^*$ and velocities $\mathbf{v}^*$ can be computed according to the positions $\mathbf{p}^n$ and velocities $\mathbf{v}^n$ from time step n:

$$\mathbf{v}^* = \mathbf{v}^n + \frac{dt}{\rho}(\mu \nabla^2 \mathbf{v}^n + \rho \mathbf{g})$$

$$\mathbf{p}^* = \mathbf{p}^n + \mathbf{v}^* dt$$

Equation 2-33: Computations of temporary position and velocity in MPS.

where dt is the time step size, μ the viscosity constant, and ρ the particle number density.

With the temporary particle positions $\mathbf{p}^*$, fluid incompressibility is violated. Then, a Poisson equation for pressure is obtained on each particle i:

$$(\nabla^2 p_i^{n+1})_i = -\frac{\rho}{dt}\frac{e_i - e^0}{e^0}$$

Equation 2-34: Poisson equation for pressure in MPS.

where p_i^{n+1} is the particle pressure. The Laplacian operator ∇^2 on pressure is defined as:

$$\nabla^2 p_i = \frac{2D}{e^0 \lambda}\sum_{j=1}^{n}(p_j - p_i) w(|\mathbf{p}_j - \mathbf{p}_i|), \quad \lambda = \frac{\int_V w(r) r^2 dv}{\int_V w(r) dv}.$$

Equation 2-35: Laplacian evaluation on pressure in MPS.

All the Poisson equations on the particles make up a system of linear equations. Once the pressures are solved, the correction velocities $\mathbf{v}'$ are computed using:

$$\mathbf{v}' = -\frac{dt}{\rho}(\nabla p^{n+1})\ .$$

Equation 2-36: Correction velocity computation in MPS.

New particle positions and velocities are finally updated for time step n+1:

$$\mathbf{v}^{n+1} = \mathbf{v}^* + \mathbf{v}'$$

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \mathbf{v}^{n+1}dt$$

Equation 2-37: Computations of new particle position and velocity in MPS.

Clavet et al. [CBP05] propose a fluid model, simplified and extended from the SPH paradigm. In this model, particle density and pressure are computed from the neighboring particles within the interaction radius, similar to the SPH-based and MPS-based models. The pseudo-density ρ_i for particle i is

$$\rho_i = \sum_{j \in N(i)} (1 - r_{ij} / h)^2$$

Equation 2-38: Pseudo-density of particle.

where $N(i)$ denotes the set of neighboring particles within the interaction radius h. And r_{ij} = $|\mathbf{p}_i - \mathbf{p}_j|$ where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the positions of particle i and j, respectively. In contrast to the SPH-based fluid density, this pseudo-density is not a true physical property. It is simply a number quantifying how the particle relates to its neighbors. Due to this contrast, this model is not classified as a SPH-based model in this dissertation. The particle pressure P_i is computed from the state equation of gas, which is the same as in [MCG03].

This model simulates the fluid incompressibility with a double density relaxation method. In this method, a new density called near-density is computed:

$$\rho_i^{near} = \sum_{j \in N(i)} (1 - r_{ij} / h)^3$$

Equation 2-39: Near-density of particle.

which uses a weight function with a sharper spike than Equation 2-38. In order to make the corresponding force exclusively repulsive, near-pressure is defined as:

$$p_i^{near} = k^{near} \rho_i^{near}$$

Equation 2-40: Near-pressure of particle.

where k^{near} is the stiff parameter. The near-pressure then produces a displacement D_{ij} for each particle pair:

$$D_{ij} = \Delta t^2 [p_i (1 - r_{ij} / h) + p_i^{near} (1 - r_{ij} / h)^2] \mathbf{r}_{ij}$$

Equation 2-41: Displacement for each particle pair.

where $\mathbf{r}_{ij}$ is the unit vector from particle i to particle j. The result of double density relaxation is a coherent fluid representation in which particles tend to be at equal distance from all immediate neighbors. Another result is that the surface tension effect is readily produced in some animations.

Furthermore, this model simulates viscoelastic fluid behavior through three sub-processes: elasticity, plasticity, and viscosity. Elasticity is obtained by inserting springs between particles, plasticity comes from the modification of the spring rest lengths, and viscosity is introduced by exchanging radial impulses determined by particle velocity differences.

Chapter 3. SPH-Based Fluid Animation

Smoothed Particle Hydrodynamics (SPH) was developed by Lucy [Luc77] and Gingold and Monaghan [GM77] for modeling astrophysical problems in three-dimensional open space. For the discussions of the SPH applications in Computational Fluid Dynamics (CFD), the interested reader is referred to the CFD literature [Mon92; LL03; Mon05]. SPH is a Lagrangian formulation and has been extended to solve a wide range of problems in CFD [BK00; GM77; Luc77; Mon94; MFZ97; Mor00; SL03]. In computer graphics, SPH is utilized to animate various materials and their phenomena, including fire and other gaseous phenomena [SF95], highly deformable bodies [DC96], lava flows [SAC*99], elastic, plastic, and melting objects [MKN*04], Newtonian fluid motions [MCG03, Roy95], fluid-solid interactions [MST*04], and fluid-fluid interactions [MSKG05; MY06].

The contributions in this dissertation are based on a SPH-based fluid modeling and animation framework. As presented in Figure 1-1, a physical-based fluid animation loop consists of the three basic steps: fluid motion computation, fluid motion integration, and fluid rendering. This SPH-based framework includes three components, corresponding to the three basic steps, which are described in sub-sections 3.1, 3.2, and 3.3, respectively.

3.1 Particle Dynamics

In the physical-based fluid animation, the Navier-Stokes (NS) equation describes the fluid motions and its terms are continuous functions such as the pressure gradient. SPH is one of the fluid modeling approaches that are able to solve the NS equation. This section discusses how the terms of the NS equation are evaluated under the SPH formulation in order to compute the fluid motions.

3.1.1 SPH Fundamentals

Under the SPH formulation, a fluid is divided into a set of elements called *particles*. The central part of SPH is an interpolation method that allows any continuous function to be expressed in terms of its values at a set of disordered particles. The interpolation method consists of two approximation steps: kernel approximation and particle approximation.

In the first approximation step, a continuous function $A(\mathbf{p})$ is expressed in an integral representation:

$$A(\mathbf{p}) = \int_{\Omega} A(\mathbf{p}')\,\delta(\mathbf{p}-\mathbf{p}')\,d\mathbf{p}'$$

Equation 3-1: Integral representation of any continuous function.

where $\mathbf{p}$ is the position at which $A(\mathbf{p})$ is evaluated, Ω the integral volume that contains $\mathbf{p}$, and $\mathbf{p}'$ any position within Ω. $\delta(\mathbf{p} - \mathbf{p}')$ is the delta function [LL03]:

$$\delta(\mathbf{p}-\mathbf{p}') = \begin{cases} 1 & \mathbf{p}=\mathbf{p}' \\ 0 & \mathbf{p}\neq\mathbf{p}' \end{cases}.$$

Equation 3-2: Delta function.

If the delta function is replaced by a kernel function $W(\mathbf{p} - \mathbf{p}', h)$, the integral representation, Equation 3-1, becomes:

$$A(\mathbf{p}) = \int_{\Omega} A(\mathbf{p}')\,W(\mathbf{p}-\mathbf{p}',h)\,d\mathbf{p}'$$

Equation 3-3: Integral representation with kernel function.

where $W(\mathbf{p} - \mathbf{p}', h)$ is the so-called kernel function, or smoothing function, or smoothing kernel, or smoothing kernel function in many SPH literatures. This dissertation chooses kernel function or kernel for short. Note that, as long as the kernel function W is not the delta function, the integral representation in Equation 3-3 can only be an approximation, and thus called kernel approximation to the function $A(\mathbf{p})$. In the kernel function, h is called smoothing length and is the radius of the spherical region centered at position $\mathbf{p}$. The further meaning of the smoothing length with respect to the kernel function W is discussed shortly in a kernel function condition.

The kernel function W should satisfy a number of conditions. The mathematical discussions for these conditions are beyond the scope of this dissertation and can be found in many SPH literatures [Mon92; LL03; Mon05]. First, the kernel function W must be continuous and differentiable. Secondly, W is usually chosen to be an even function such that:

$$W(\mathbf{p} - \mathbf{p}', h) = W(\mathbf{p}' - \mathbf{p}, h).$$

Equation 3-4: Kernel function as even function.

Given this condition, the kernel approximation would have h^2 accuracy. The third condition is the normalization condition that states:

$$\int_\Omega W(\mathbf{p} - \mathbf{p}', h) d\mathbf{p}' = 1$$

Equation 3-5: Normalization condition.

where the integration is taken over the spherical region centered at position $\mathbf{p}$ with radius h. The fourth condition is the delta function property:

$$\lim_{h \to 0} W(\mathbf{p} - \mathbf{p}', h) = \delta(\mathbf{p} - \mathbf{p}').$$

Equation 3-6: Delta function property.

The fifth condition is the compact condition:

$$W(\mathbf{p}-\mathbf{p}',h)=0 \text{ when } |\mathbf{p}-\mathbf{p}'|>h$$

Equation 3-7: Compact condition.

which states that the kernel function *W* is effective (non-zero) within the spherical region of radius *h* centered at position **p**. The effective region is called the support domain for position **p**. Given this condition, the integration over the entire problem domain in Equation 3-3 is localized as the integration over the support domain which is often much smaller and thus more efficient for the computation. Under the above mentioned five conditions, different kernel functions can be designed for different purposes. The kernel function *W* and the smoothing length *h* are further discussed in sub-sections 3.1.3 and 3.1.4, respectively.

The second approximation step in the SPH interpolation method is the particle approximation. At this step, the kernel approximation is further approximated by a discrete form of summation. That is, the continuous integral representation in Equation 3-3 is replaced by the discrete summation over all the particles in the support domain, and the infinitesimal volume $d\mathbf{p}'$ is replaced by the finite volume of the particle ΔV_j:

$$A(\mathbf{p})=\sum_{j=1}^{n} A(\mathbf{p}_j)W(\mathbf{p}-\mathbf{p}_j,h)\Delta V_j$$

Equation 3-8: Discrete summation to replace integral representation.

where *n* is the number of the particles within the support domain for position **p**, *j* the particle index, and $\mathbf{p}_j$ the particle position. The finite particle volume can be expressed as:

$$\Delta V_j = \frac{m_j}{\rho_j}$$

Equation 3-9: Finite particle volume.

where m_j is the particle mass and ρ_j the particle density. Thus, the continuous function $A(\mathbf{p})$ is approximated by the discrete particle summation:

$$A(\mathbf{p}) = \sum_{j=1}^{n} \frac{m_j}{\rho_j} A(\mathbf{p}_j) W(\mathbf{p} - \mathbf{p}_j, h).$$

Equation 3-10: Particle approximation to continuous function.

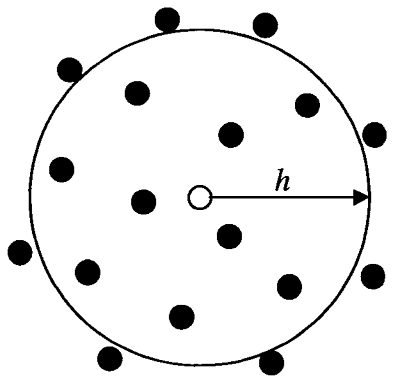


Figure 3-1: Particle approximation in support domain.

Equation 3-10 is the particle approximation and is practically an equation to compute the average fluid attributes. Specifically, the value of function $A(\mathbf{p})$ at position $\mathbf{p}$ is approximated using the average of the function values at the neighboring particles. The kernel function W is then a weight function. The weight is computed from the distance between a neighboring particle to position $\mathbf{p}$, and it determines how much the value $A(\mathbf{p}')$ at the neighboring particle contributes to $A(\mathbf{p})$. The particle approximation is illustrated in Figure 3-1, where the hollow dot denotes position $\mathbf{p}$ at which the function $A(\mathbf{p})$ is evaluated, the big circle the support domain for position **p**, *h* the radius of the support domain, and the black dots denote the neighboring particles. Some of the particles are within the support domain.

Following the same method to approximate the continuous function *A*(**p**), the particle approximation to the gradient of *A*(**p**) can be obtained as:

$$\nabla A(\mathbf{p}) = \sum_{j=1}^{n} \frac{m_j}{\rho_j} A(\mathbf{p}_j) \nabla W(\mathbf{p} - \mathbf{p}_j, h) .$$

Equation 3-11: Particle approximation to gradient of continuous function.

Similarly, the Laplacian of *A*(**p**) can be approximated by:

$$\nabla^2 A(\mathbf{p}) = \sum_{j=1}^{n} \frac{m_j}{\rho_j} A(\mathbf{p}_j) \nabla^2 W(\mathbf{p} - \mathbf{p}_j, h) .$$

Equation 3-12: Particle approximation to Laplacian of continous function.

In both Equation 3-11 and Equation 3-12, the gradient and the Laplacian of $W(\mathbf{p} - \mathbf{p}_j, h)$ are evaluated with respect to **p**. The essential idea is that the differentiation of a function is approximated from the function values at particles and the differentiation of the kernel function. This is one reason for the kernel function to be differentiable. Note that no specific particle configuration is required and the particles can be arbitrarily distributed within the support domain. Thus, there is no need to use a grid or mesh for the differentiation.

In the particle approximation, the particle mass and density are introduced into the equations. These two fluid attributes naturally exist for a small piece of the fluid which is the fluid element in SPH.

3.1.2 Particle Motion Computation

In SPH, a fluid is modeled as an aggregation of a number of particles. Fluid motion is characterized by all particle motions. Each particle motion is governed by the Lagrangian version of the Navier-Stokes (NS) equation:

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla P_i + \mu\nabla^2\mathbf{v}_i + \mathbf{f}_i$$

Equation 3-13: Lagrangian version of NS equation for particle.

where i is the particle index, $\mathbf{v}_i$ the velocity of particle i, ρ_i the particle density, P_i the particle pressure, μ the viscosity, and $\mathbf{f}_i$ the summation of external forces on particle i such as gravity. The terms in Equation 3-13 are continuous functions. According to the SPH fundamentals, they can be computed in terms of the neighboring particles in the support domain for particle i.

At first, the particle density ρ_i is computed according to Equation 3-10:

$$\rho_i = \sum_{j=1}^{n} \frac{m_j}{\rho_j}\rho_j W(\mathbf{p}_i - \mathbf{p}_j, h) = \sum_{j=1}^{n} m_j W(\mathbf{p}_i - \mathbf{p}_j, h)$$

Equation 3-14: Particle density.

where $\mathbf{p}_i$ is the position of particle i, and m_i and m_j the particle masses for particle i and j, respectively. Equation 3-14 simply states that the density of a particle is a weighted average of the densities of all the particles in its support domain. Given the particle density, the particle pressure P_i can be computed using the equation of state [DC96]:

$$P_i = K(\rho_i - \rho_0)$$

Equation 3-15: Particle pressure from equation of state.

where K is the gas constant and ρ_0 the original fluid density. Equation 3-15 is the ideal gas state equation for compressible fluid. It is called a weakly incompressible method to model incompressible fluid with Equation 3-15. This method is adopted by the previous graphical fluid model [MCG03].

In SPH, there are different ways to evaluate the pressure gradient term and the viscosity term in Equation 3-13. The following evaluations are proposed by Muller et al. [MCG03] and are widely used in computer graphics as well as in this dissertation:

$$-\frac{1}{\rho_i}\nabla P_i = -\frac{1}{\rho_i}\sum_{j=1}^{n}\frac{(P_i + P_j)}{2}\frac{m_j}{\rho_j}\nabla W(\mathbf{p}_i - \mathbf{p}_j, h)$$

Equation 3-16: Particle pressure gradient.

$$\mu\nabla^2\mathbf{v}_i = \mu\sum_{j=1}^{n}(\mathbf{v}_j - \mathbf{v}_i)\frac{m_j}{\rho_j}\nabla^2 W(\mathbf{p}_i - \mathbf{p}_j, h)$$

Equation 3-17: Particle viscosity.

where the gradient and the Laplacian of the kernel function are taken with respect to $\mathbf{p}_i$. After computing equations Equation 3-14 to Equation 3-17, the particle acceleration can be computed according to Equation 3-13. ($\mathbf{f}_i$ can be computed trivially or obtained from the interactions with other animation identities [MCG03].) Then, the particle velocity and position can be computed using numerical integration.

3.1.3 Kernel Function

The kernel function $W(\mathbf{p}_i - \mathbf{p}_j, h)$ computes the weight of particle j in a weighted average in order to approximate a function value at position $\mathbf{p}_i$ or at particle i of position $\mathbf{p}_i$. In this dissertation, the latter case is followed by default unless indicated explicitly. In addition to the five kernel conditions (described in the sub-section of SPH fundamentals), the kernel function is typically a function of the distance $r_{ij} = |\mathbf{p}_i - \mathbf{p}_j|$ between the two particles. For short in this dissertation, the kernel function is expressed as $W(|\mathbf{p}_i - \mathbf{p}_j|, h)$, or $W(r_{ij}, h)$, or W_{ij}. Then, the gradient of the kernel function W_{ij} with respect to position $\mathbf{p}_i$ can be expressed as:

$$\nabla_i W_{ij} = \frac{\mathbf{p}_i - \mathbf{p}_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} = \frac{\mathbf{p}_{ij}}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} = \mathbf{n}_{ij} \frac{\partial W_{ij}}{\partial r_{ij}},$$

Equation 3-18: Gradient of kernel function with respect to one position.

where $\mathbf{p}_{ij} = \mathbf{p}_i - \mathbf{p}_j$ and $\mathbf{n}_{ij}$ is a normalized vector: $\mathbf{n}_{ij} = \mathbf{p}_{ij}/|\mathbf{p}_{ij}|$. According to Equation 3-18, it is straightforward to have the gradient of W_{ij} with respect to the other position $\mathbf{p}_j$:

$$\nabla_j W_{ij} = -\nabla_i W_{ij} .$$

Equation 3-19: Gradient of kernel function with respect to other position.

The Laplacian of the kernel function W_{ij} with respect to position $\mathbf{p}_i$ is:

$$\begin{aligned} \nabla_i^2 W_{ij} &= \frac{\partial(\nabla_i W_{ij})_x}{\partial p_{ix}} + \frac{\partial(\nabla_i W_{ij})_y}{\partial p_{iy}} + \frac{\partial(\nabla_i W_{ij})_z}{\partial p_{iz}} \\ &= \frac{2}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}} + \frac{\partial^2 W_{ij}}{\partial r_{ij}^2} \end{aligned}$$

Equation 3-20: Laplacian of kernel function with respect to one position.

which is equal to the one with respect to position $\mathbf{p}_j$:

$$\nabla_j^2 W_{ij} = \nabla_i^2 W_{ij} .$$

Equation 3-21: Gradient of kernel function with respect to another position.

Different kernel functions can be designed for different purposes to evaluate fluid attributes, as long as the kernel function satisfies the five kernel conditions. One advantage of SPH is that the kernel function W_{ij} can be computed in a programming subroutine, and then it is straightforward to change the code for one kernel function into the code for another [Mon92].

The traditional kernel function from [Mon92] is

$$W_{spline}(r_{ij}, h) = \frac{1}{\pi h^3} \begin{cases} 1 - 1.5q^2 + 0.75q^3 & 0 \le q \le 1 \\ 0.25(2-q)^3 & 1 \le q \le 2 \\ 0 & otherwise \end{cases}$$

where $q = \frac{2r_{ij}}{h}$.

Equation 3-22: Traditional spline kernel function.

The choice of the kernel functions largely influences the stability, accuracy, and speed of a SPH-based fluid model. Muller et al. [MCG03] design the following kernel function for their interactive fluid animations:

$$W_{poly6}(r_{ij}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r_{ij}^2)^3 & 0 \le r_{ij} \le h \\ 0 & otherwise \end{cases}.$$

Equation 3-23: Polynomial kernel function for interactive fluid animation.

where r_{ij} only appears squared so that the efficiency is improved by avoiding the computation of square roots. However, if the kernel function W_{poly6} is used in Equation 3-16 to compute the pressure gradient, the particles tend to cluster. This is because, when two particles get very close to each other, the pressure force between them vanishes. The following is the gradient of the kernel function W_{poly6} with respect to particle i:

$$\nabla_i W_{poly6}(r_{ij}, h) = \mathbf{n}_{ij} \frac{315}{64\pi h^9} \begin{cases} -6r_{ij}(h^2 - r_{ij}^2)^2 & 0 \le r_{ij} \le h \\ 0 & otherwise \end{cases}.$$

Equation 3-24: Gradient of polynomial kernel function.

It can be seen from Equation 3-24 that, as the particle distance r_{ij} decreases, the magnitude of the gradient gets smaller and smaller. To avoid the vanishing pressure force, Desbrun and Cani [DC96] propose a spiky kernel function:

$$W_{spiky}(r_{ij},h) = \frac{15}{\pi h^6}\begin{cases}(h-r_{ij})^3 & 0 \le r_{ij} \le h \\ 0 & otherwise\end{cases}$$

Equation 3-25: Spiky kernel function for repulsive pressure forces.

and the gradient of the spiky kernel function W_{spiky} with respect to particle i is:

$$\nabla_i W_{spiky}(r_{ij},h) = \mathbf{n}_{ij}\frac{15}{\pi h^6}\begin{cases}-3(h-r_{ij})^2 & 0 \le r_{ij} \le h \\ 0 & otherwise\end{cases}.$$

Equation 3-26: Gradient of spiky kernel function for repulsive pressure forces.

It can be seen from Equation 3-26 that, as the particle distance r_{ij} decreases, the magnitude of the gradient increases and approaches a constant. Thus, the spiky kernel function would not result in the vanishing pressure forces for very close particles. The shorter the particle distance, the bigger the repulsive force is.

Viscosity is a resistance to changing the shape of a fluid and is caused by the internal friction of the fluid. In SPH, viscosity smoothes the velocity between particles, that is, decreases the particle relative velocity. In the last sub-section, the viscosity is formulated in Equation 3-17. If a standard kernel function is used to compute the viscosity, the particle relative velocity may be decreased for some particle distances but increased for the others. For example, the Laplacian of the kernel function W_{poly6} is as follows:

$$\nabla_i^2 W_{poly6}(r_{ij},h) = \frac{315}{64\pi h^9}\begin{cases}6(h^2-r_{ij}{}^2)(7r_{ij}{}^2-3h^2) & 0 \le r_{ij} \le h \\ 0 & otherwise\end{cases}.$$

Equation 3-27: Laplacian of polynomial kernel function.

According to Equation 3-17, the particle relative velocity for particle i with respect to particle j, $\mathbf{v}_i - \mathbf{v}_j$, would be decreased when $r_{ij} > \sqrt{3/7}\,h$; but increased when $r_{ij} < \sqrt{3/7}\,h$. To model the constant of the velocity smoothing, Muller et al. [MCG03] propose a new kernel function $W_{viscosity}(r_{ij}, h)$ for evaluating the viscosity term in the NS equation:

$$W_{viscosity}(r_{ij},h) = \frac{15}{2\pi h^3}\begin{cases} -\dfrac{r_{ij}^3}{2h^3} + \dfrac{r_{ij}^2}{h^2} + \dfrac{h}{2r_{ij}} - 1 & 0 \le r_{ij} \le h \\ 0 & otherwise \end{cases}.$$

Equation 3-28: Viscosity kernel function.

The Laplacian of this kernel function is:

$$\nabla_i^2 W_{viscosity}(r_{ij},h) = \frac{45}{\pi h^6}\begin{cases} (h - r_{ij}) & 0 \le r_{ij} \le h \\ 0 & otherwise \end{cases}$$

Equation 3-29: Viscosity kernel function.

which does not result in the increase of the particle relative velocity with Equation 3-17.

3.1.4 Smoothing Length

As indicated in [Mon92; LL03], the kernel approximation error is normally $O(h^2)$. This is confirmed by numerical analysis [BK00]. However, as reported in Schlatter [Sch99], if the smoothing length h is too small, then the particles may have too few neighbors and the statistical errors are high for the particle approximations in Equation 3-10 to Equation 3-12. By now, the optimal smoothing length h has not been found for generic SPH applications.

The smoothing length h affects the accuracy and the efficiency of the computed results. In physical-based fluid animation, the accuracy is not evaluated with numerical analysis as is performed in CFD. As indicated in the introduction chapter, the animated fluid motions are evaluated by human observers with typical knowledge of fluids. Desbrun and Cani [DC96] point out that a small value of the smoothing length h will create very local particle interactions so that the animated fluids will separate easily into pieces. This observation is consistent with the experimental results generated for this dissertation.

Besides accuracy, the efficiency is also affected by the smoothing length h. The complexity of a SPH-based particle system is usually $O(nN)$, where N is the total number of particles and n the average number of neighbors of each particle. The bigger the smoothing length h, the larger the n and thus the higher the complexity. Muller et al. [MCG03] propose a SPH-based model for interactive fluid animation. They demonstrate a system with an interactive fluid animation at 5 frames per second. The animation consists of 3000 particles. Unfortunately, the average number of neighbors for each particle is not given in their model. Neither is in another SPH-based model [CBP05] which also achieves interactive rate for fluid animation. In another Muller's SPH-based model for the animation of elastic, plastic and melting objects [MKN*04], 10 nearest neighbors are used in the SPH interpolations (Equation 3-10 to Equation 3-12) and an interactive animation speed is achieved.

In CFD, Cummins and Rudman [CR99] propose that the smoothing length h is chosen to be a constant and to be 1 to 1.5 times the original particle distance depending on the applications. In their work, the particles are originally distributed on a 2D grid, and the simulated fluids are confined within a closed rectangular boundary without free surfaces. In physical-based fluid animation, however, the SPH particles are distributed in 3D and the animated fluids usually have free surfaces. To make the trade-off between accuracy and efficiency, the value of the smoothing length h is experimented from 1 to 3 times the original particle distance. The value of 2 times the original particle distance results in the better trade-off than others, and thus is chosen for the smoothing length h in this dissertation. Accordingly, the number of the neighbors for a particle fluctuates around 30 in the produced animations.

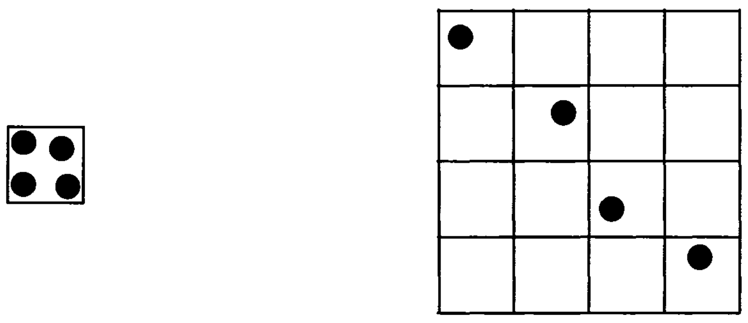
3.1.5 Neighbor Search

In a SPH-based model, it is an inevitable frequent operation to find neighbors for each particle. The naïve method is to check all the particles for each particle. The computational complexity of such a method is $O(N^2)$ where N is the number of the total particles. The proposed model adopts a more efficient searching method using a grid-based data structure. A bounding box enclosing all the particles is created and subdivided into uniform cubic cells. The side length of each cell is equal to the smoothing length h. All the particles are each registered to the cells that enclose them. The neighbors of a given particle in a cell are either in the same cell as the given particle or in the 26 surrounding cells. A search for the neighbors is then performed in these 27 cells. The computational complexity is $O(mN)$, where m is the average number of particles in each search and $m \geq n$ but is much smaller than N, and n is the average number of neighbors of each particle. The neighbor search for each particle is performed at every time step because of particle movements.

To further improve the efficiency, the uniform cubic cells may be instantiated and stored in a hash table only when they are not empty; or, the octree structure may be used for the neighbor search. These improvements are necessary for the interactive models.

Since this dissertation is not focused on interactive fluid animation, the simple uniform cubic cell is adopted for the neighbor search.

A further observation about the neighbor search is that the actual searching performance is also dependent on the actual fluid deformation. For example, if a fluid is aggregating within a spherical region, then the data structure for the neighbor search would be much smaller than the one for a spreading fluid. An example of the data structure in uniform cells is illustrated in Figure 3-2, where only 1 cell is needed in (a) but 16 needed in (b) in order to cover 4 fluid particles (denoted by 4 black dots). Similar examples can also be constructed for the octree structure.



(a) (b)

Figure 3-2: Particle neighbor search dependent on fluid deformation.

3.1.6 Surface Tension

Surface tension is an effect within the surface layer of a fluid that causes the layer to behave like as an elastic sheet. It is a critical factor in many fluid phenomena such as capillary motion and a coin being prevented from sinking. Surface tension is caused by the attraction between the molecules of the fluid. In the fluid interior, each molecule is pulled equally in all directions by its neighboring molecules so that the resulting force is zero.

On the other hand, at the surface of the fluid, the molecules are pulled inwards by other molecules deeper inside the fluid while there are no fluid molecules from the outside. As a result, all the surface molecules are subject to a resulting force directed towards the fluid interior. The inward resulting force would squeeze the fluid until it is balanced by the fluid resistance to compression. Thus, the fluid tends to have the smallest surface area possible. Mathematically, the fluid tries to minimize the surface curvature. The larger the curvature, the higher the surface tension force. Surface tension appears not only on the fluid-air interface but also on the interface between two fluids. Thus, the surface tension force also depends on the tension coefficient σ which is associated with the two fluids forming the interface.

Surface tension is simulated with a color field in the previous SPH-based fluid model [MCG03]. The color field c is defined as:

$$c(\mathbf{p}) = \sum_{j=1}^{n} \frac{m_j}{\rho_j} W(\mathbf{p}-\mathbf{p}_j, h)$$

Equation 3-30: Color field function.

where the symbols follow the same meanings as in the conventional SPH equations such as Equation 3-10. Equation 3-30 computes the *color* value at position **p** in the animation space. The surface normal field **n** is then defined as the gradient of the color field:

$$\mathbf{n} = \nabla c\ .$$

Equation 3-31: Surface normal defined as gradient of color field.

The divergence of **n** measures the surface curvature κ:

$$\kappa = \frac{-\nabla^2 c}{|\mathbf{n}|}$$

Equation 3-32: Surface curvature defined as divergence of surface normal.

where the minus sign is required to get positive curvature for convex fluid volumes. The surface tension force $\mathbf{f}_s$ is computed as:

$$\mathbf{f}_s = \sigma\kappa\mathbf{n} = -\sigma\nabla^2 c \frac{\mathbf{n}}{|\mathbf{n}|}.$$

Equation 3-33: Surface tension force computed from color field.

Evaluating **n**/|**n**| at positions where |**n**| is very small causes numerical problems. Thus, the surface tension force is computed only if |**n**| exceeds a threshold value. This condition can also be used to identify the surface region.

3.1.7 Interactions with Solids

Many fluid phenomena involve fluid-solid interactions. For example, pouring water into a glass involves the water interacting with the glass. Two previous particle-based fluid models [MST*04; CBP05] present the modeling of fluid-solid interactions, which can be easily adopted by a SPH-based fluid model.

Muller et al. [MST*04] model the fluid interactions with deformable solids. The solid object is represented by a mesh and the mesh nodes carry the displacements, velocities and forces. The fluid is represented by a set of particles, and the fluid particles carry the positions, velocities and internal forces. The interaction occurs at the interface between the fluid and the solid. Three interface conditions are formulated to model the interaction. The first condition is the no-penetration condition, which is described by the following equation:

$$(\frac{\partial \mathbf{u}}{\partial t} - \mathbf{v}) \cdot \mathbf{n} = 0$$

Equation 3-34: No-penetration condition.

where **u** is the solid deformation rate, **v** the fluid velocity, and **n** the interface normal. The equation states that the fluid velocity and the solid deformation rate are equal to each other along the interface normal. The no-penetration condition prevents the fluid from penetrating into the solid. The second condition is the no-slip condition:

$$(\frac{\partial \mathbf{u}}{\partial t} - \mathbf{v}) \times \mathbf{n} = 0$$

Equation 3-35: No-slip condition.

which holds for most fluid-solid interfaces and states that the fluid velocity and the solid deformation rate are equal to each other along the interface tangential direction. The two conditions can be combined into a more elegant form:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{v} .$$

Equation 3-36: No-penetration and no-slip conditions.

The third condition is the reaction condition:

$$\sigma_s \mathbf{n} = \sigma_f (-\mathbf{n})$$

Equation 3-37: Reaction condition.

where **n** is the interface normal pointing from the solid to the fluid, σ_s the solid stress, and σ_f the fluid stress. These three conditions are approximated with particle-based equations which are not used in this dissertation and thus are omitted. The interested reader is referred to the original paper for detail [MST*04].

Clavet et al. [CBP05] model the collisions between fluids and rigid bodies. A collision is identified if a fluid particle is too close to or is inside a rigid body. The collision processing consists of three steps. In the first step, the impulses due to the particle-rigid body collisions are accumulated into force and torque buffers. In the second step, the rigid bodies are advanced by the accumulated force and torque. In the final step, the particles are advanced by the collision impulses. The impulse due to a particle-rigid body collision is computed as follows. At first, the collision velocity $\mathbf{v}_c$ is computed as:

$$\mathbf{v}_c = \mathbf{v}_i - \mathbf{v}_r$$

Equation 3-38: Collision velocity.

where $\mathbf{v}_i$ is the particle velocity and $\mathbf{v}_r$ the rigid-body velocity at the collision point. Then, the collision velocity is decomposed into two components:

$$\mathbf{v}_n = (\mathbf{v}_c \cdot \mathbf{n})\mathbf{n}$$

$$\mathbf{v}_t = \mathbf{v}_c - \mathbf{v}_n$$

Equation 3-39: Components of collision velocity.

where $\mathbf{n}$ is the rigid-body surface normal at the collision point, $\mathbf{v}_n$ the normal component, and $\mathbf{v}_t$ the tangential component. At last, the collision impulse $\mathbf{I}$ is:

$$\mathbf{I} = \mathbf{v}_n - \alpha\,\mathbf{v}_t$$

Equation 3-40: Collision impulse.

where α is a friction parameter enabling slip ($\alpha = 0$) or no-slip ($\alpha = 1$) surface conditions.

In this dissertation, the solids only play a supporting role in the example animations which are mainly intended to demonstrate the non-Newtonian fluid behaviors. Thus, the fluid-rigid body interaction model [CBP05] is adopted.

3.2 Motion Integration

Several motion integration methods are used in the previous particle-based fluid models, and are also applicable to a SPH-based fluid model. These methods are: the Euler method, the leap-frog method, the prediction-relaxation method, and the prediction-correction method.

The Euler method is computationally inexpensive and is easy to implement. The particle velocity **v** and position **p** are integrated as follows:

$$\mathbf{v}^{n+1} = \mathbf{v}^{n} + \mathbf{a}^{n}\Delta t$$

$$\mathbf{p}^{n+1} = \mathbf{p}^{n} + \mathbf{v}^{n+1}\Delta t$$

Equation 3-41: The Euler integration method.

where the supersript n denotes the nth time step, Δt the time step, and **a** the particle acceleration which is computed from the particle dynamics. The Euler method is adopted in this dissertation because of its simplicity.

In comparison to the Euler method, the leap-frog method gives higher order accuracy and allows larger time steps for the stable animation [MCG03; MSKG05]. The particle velocity **v** and position **p** are integrated as follows:

$$\mathbf{v}^{n+1/2} = \mathbf{v}^{n-1/2} + \mathbf{a}^{n}\Delta t$$

$$\mathbf{p}^{n+1} = \mathbf{p}^{n} + \mathbf{v}^{n+1/2}\Delta t$$

Equation 3-42: Leap-frog integration method.

where the superscript $n + 1/2$ and $n - ½$ denote the middle points of the time steps. The velocity $\mathbf{v}^n$ at the nth time step for computing $\mathbf{a}^n$ is estimated as follows:

$$\mathbf{v}^n = \mathbf{v}^{n-1/2} + \mathbf{a}^n \frac{\Delta t}{2}.$$

Equation 3-43: Velocity estimation.

Clavet et al. [CBP05] propose a prediction-relaxation integration method which consists of two steps. First, the particle velocity **v** is updated according to gravity and viscosity:

$$\mathbf{v} = \mathbf{v} + \Delta t(\mathbf{g} + \mathbf{a}_v)$$

Equation 3-44: Velocity update at prediction step.

where **g** is the gravity and $\mathbf{a}_v$ the viscosity acceleration. Then, the particle position **p** is saved as the previous position $\mathbf{p}_{prev}$, and the particle is moved according to the updated velocity **v**:

$$\mathbf{p}_{prev} = \mathbf{p}$$

$$\mathbf{p} = \mathbf{p} + \Delta t\,\mathbf{v}.$$

Equation 3-45: Particle movement to predicted position.

This step is called prediction step because the particle is moved to a predicted position.

In the second step, the particle position is further moved according to the spring force and other constraint forces such that volume conservation, anti-clustering, and surface tension are enforced:

$$\mathbf{p} = \mathbf{p} + \Delta t^2\,\mathbf{f}_1 / m$$

$$\mathbf{p} = \mathbf{p} + \Delta t^2\,\mathbf{f}_2 / m$$

$$\cdots$$

$$\mathbf{p} = \mathbf{p} + \Delta t^2\,\mathbf{f}_n / m$$

Equation 3-46: Particle relaxation from predicted position.

where $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$ are the constraint forces and m is the particle mass. Finally, the particle velocity is recomputed from the saved previous particle position and the relaxed particle position:

$$\mathbf{v} = (\mathbf{p} - \mathbf{p}_{prev})/\Delta t \ .$$

Equation 3-47: Particle velocity re-computation.

If only one constraint force is used, the integration can be simplified to the form similar to the usual leap-frog method. The sequential application of the constraint forces is analogous to operator splitting often used in the grid-based stable fluid simulation [Sta99] and can lead to better stability than the leap-frog method.

The fluid volume conservation in this model [CBP05] could be improved with the prediction-correction integration method which is presented in the MPS-based fluid model [PTB*03]. In this dissertation, a SPH-based prediction-correction integration method is presented in the first contribution in Chapter 4.

3.3 Fluid Rendering

Fluids are material volumes. To render fluids, there are two techniques: volume rendering and surface rendering. Volume rendering is the direct visualization of a three dimensional volume, without the use of an intermediate surface representation. Surface rendering is the visualization of a surface representation, either an explicit surface mesh or an implicit surface, that encloses a three dimensional volume.

In physical-based fluid animation, surface rendering is typically faster than volume rendering. One disadvantage of surface rendering for fluids is that the surface representation does not have the fluid interior information. However, the animated fluids so far usually have uniformed interior properties, and the fluid interior illumination can be approximately simulated by ray-tracing techniques. Therefore, surface rendering is more popular than volume rendering.

3.3.1 Implicit Surface

Fluids have ever changing shapes. If their surfaces are represented by explicit surface meshes, it is very difficult as well as expensive to maintain the mesh quality in continuous animations. Thus, a popular approach to fluid surface representation is the use of implicit surfaces (or iso-surfaces). As the name suggests, implicit surfaces are not made of the explicit geometric or parametric primitives. Instead, they are implicitly defined by functions.

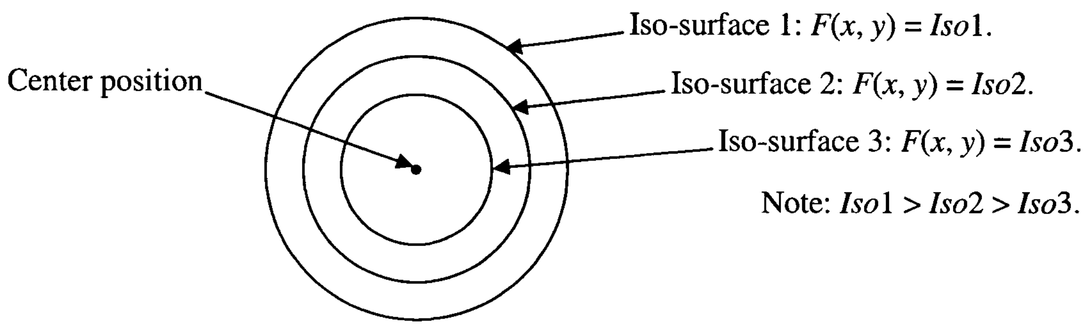

Iso-surface 1: $F(x, y) = Iso1$.
Center position
Iso-surface 2: $F(x, y) = Iso2$.
Iso-surface 3: $F(x, y) = Iso3$.
Note: $Iso1 > Iso2 > Iso3$.

Figure 3-3: Iso-surfaces with different iso-values.

Consider the function $F(x, y)$: $x^2 + y^2 = r^2$. It can represent an infinite number of 2D points (x, y) that lie on the circumference of a circle which has radius r and is centered at the origin of a 2D Cartesian coordinate system. The circle is implicitly represented by the function. The same thing can be done in 3D where the function $F(x, y, z)$: $x^2+y^2+z^2 = r^2$

represents all the 3D points (x, y, z) on the surface of a sphere of radius r. The function of the sphere can be rewritten as: $F(x, y, z) = Iso$, where $Iso = r^2$ and is called iso-value. This is why implicit surfaces are also called iso-surfaces. Now, by changing the iso-value, different surfaces, depending on the function F, can be represented, which are illustrated in Figure 3-3 using the above circle function. The function F is called implicit function or scalar field function. The center point is the controlling point as opposed to the surface points. Different function F can be used to describe the field around a controlling point, and more than one controlling points can be specified such that the resulting surface is a complex blend of the simpler spherical surfaces. With the implicit function, it is easy to determine if a point is on, inside of, or outside of an implicit surface. Assume that point a is at position (x_a, y_a, z_a). This point is on the implicit surface if $F(x_a, y_a, z_a) = Iso$; inside if $F(x_a, y_a, z_a) > Iso$; and outside if $F(x_a, y_a, z_a) < Iso$. A comprehensive introduction to implicit surfaces can be found in [Blo97].

Implicit surfaces are widely used to animate fluids that involve large changes in shape and topology. In particle-based fluid models, implicit surfaces are computed based on the particle positions, i.e. particles are the controlling points to implicit surfaces.

3.3.2 Implicit Surface in SPH

SPH provides a natural way of defining fluid implicit surfaces. The SPH particle density Equation 3-14 is applicable not only to the particle positions but also to any positions in the animation space. Since the kernel function W is smoothly defined, the density distribution is continuous in the animation space. In the region occupied by a fluid, the density is larger than zero. And, in the empty region, the density is zero. Therefore, a non-zero iso-density ρ_{iso} defines the implicit surface of the fluid, and the particle density Equation 3-14 is the implicit function. This implicit surface is used in this dissertation.

Now, the question is how to choose an appropriate iso-density ρ_{iso} as the iso-value for the implicit function. The idea of using iso-density to define implicit surfaces is first proposed in [DC96], where an iso-density is derived from the two-particle case. Based on the same idea, a reasonable range between $\rho_{\min}$ and $\rho_{\max}$ is decided such that

$$\rho_{\min} < \rho_{\text{iso}} < \rho_{\max}$$

Equation 3-48: Reasonable iso-density range.

where

$$\rho_{\min} = 0$$

Equation 3-49: Lower limit of iso-density range.

and

$$\rho_{\max} = mW(0,h) .$$

Equation 3-50: Upper limit of iso-density range.

In Equation 3-50, m is the particle mass, W the kernel function, and h the smoothing length. The value of $\rho_{\min}$ is easily justified because non-positive density is not physically possible. The value of $\rho_{\max}$ is the density at the position of a stand-alone particle that does not have any effective neighbors. $\rho_{\max}$ is the upper limit because, if $\rho_{\text{iso}} \geq \rho_{\max}$, then a stand-alone particle would be on or outside of the implicit surface and thus could not be rendered in the animation. This contradicts the SPH assumption that a particle is a piece of fluid and thus should be visible in the animation. Once the iso-density range is determined, different values of ρ_{iso} can be tested to see if the corresponding iso-surfaces closely fit with the aggregation of the particles. According to the experiments for this dissertation, it seems that the values around the middle of the range give better results than those close to the two ends.

Implicit surfaces of a fluid can also be defined by a color field function. In this case, a color field value is chosen as the iso-value. A couple of the color field functions are available. The first one $c(\mathbf{p})$ is defined in Equation 3-30, proposed by Muller et al. [MCG03]. The second one $\varphi(\mathbf{p})$ is proposed by Clavet et al. [CBP05] and is defined as:

$$\varphi(\mathbf{p}) = \left(\sum_{j \in N(\mathbf{p})} \left(1 - \frac{|\mathbf{p} - \mathbf{p}_j|}{h} \right)^2 \right)^{\frac{1}{2}}$$

Equation 3-51: Another color field function.

where $N(\mathbf{p})$ denotes the set of the neighboring particles within the radius h to position $\mathbf{p}$ and $\mathbf{p}_j$ the position of particle j.

3.3.3 Implicit Surface Rendering

Usually, there are two strategies to render an implicit surface with good quality. The first is to ray-trace the implicit surface directly. This strategy is fast for simple implicit surfaces [MP89; DC98] but is quite slow for complex ones [FM96; FF01; EMF02; PTB*03]. The second is to ray-trace a triangular mesh that approximates the implicit surface. This strategy provides good rendering quality at a reasonable speed by taking advantage of computer graphics hardware. Therefore, it is used by many fluid models [WMW86; Roy95; HK03; MCG03; TFK*03; CBP05; MY06] as well as in this dissertation. In these fluid models, the triangular mesh is acquired using the Marching Cube algorithm [LC87] or its variants, and then is ray-traced by the rendering software to produce the image. In this research, the free rendering software, POVRay, which is publicly available at: http://povray.org/, is used.

Chapter 4. Particle-Based Non-Newtonian Fluids

4.1 Background

Viscoelastic fluids are examples of non-Newtonian fluids. In computer graphics, one of the earliest viscoelastic models is proposed in [TF88], and two recent ones in [GBoB04; CBP05]. In this chapter, a new particle-based model for the animations of non-Newtonian fluids is presented. The new model includes two new methods. The first is a new particle dynamics method for non-Newtonian fluids. In the previous particle-based non-Newtonian fluid models [TF88; CBP05], the stress tensor is based on a linear combination of elastic spring, viscous dashpot, and the von Mises yield condition, which can only model linear non-Newtonian fluid motions. Meanwhile, the previous grid-based non-Newtonian model fluid [GBoB04] computes the stress tensor based on a quasi-linear Maxwell model and the von Mises yield condition. This model violates frame indifference (or frame invariance as in some CFD literature) when the fluids are in rotational motions. In contrast, the new particle dynamics method is based on a non-linear Maxwell model [EKH02], and takes into account frame indifference for arbitrary fluid motions. An experimental result is presented later to demonstrate that the new particle dynamics method is more accurate in realistically animating non-Newtonian fluid phenomena. In addition, non-Newtonian fluids are incompressible. To enforce the fluid incompressibility, the Poisson equation for pressure is more suitable than the state equation of gas [CBP05]. The latter is used in the previous SPH-based fluid models [DC96; SAC*99; MCG03; CBP05], and the former is in other fluid models [Sta99; FF01; PTB*03]. In the new particle dynamics method, the Poisson equation for pressure is applied within the SPH framework.

The second new method is a particle re-sampling method. In the particle-based fluid models, particles are distributed in a fluid. The flowing and deformation of the fluid can cluster particles in some regions and spread them in others. Since the fluid attributes are computed from the particle distribution in the neighborhood, the computation is not accurate in the under-sampled particle regions. Consequently, particles may form unrealistic clusters in the animation results. To tackle the particle clustering problem, Desbrun and Cani [DC96] propose a repulsive pressure force between particles such that particles do not get too close to each other. And, Clavet et al. [CBP05] propose a similar anti-clustering method using repulsive pressure force. Both of these methods handle the over-sampled particle regions very well, but have little to do with the under-sampled particle regions. Desbrun and Cani [DC99] propose a particle re-sampling method in which particles are re-sampled according to the particle pressure variations. This re-sampling method is applicable for adaptive space discretization and is not able to maintain a good distribution of the particles. Based on that re-sampling method [DC99], Pauly et al. [PKA*05] try to deal with the "poor spatial discretization" of the particles. Their criterion of particle under-sampling is that, if a particle has 10 or less neighboring particles, then its spherical neighborhood is an under-sampled region; otherwise not. This criterion would miss some under-sampling cases. For example, if 11 neighboring particles are crowded in one half of the spherical neighborhood and the other half is empty, then the empty half is under-sampled and the spherical neighborhood is a poor spatial discretization but would not be detected. Conveniently in their model, when the under-sampled regions are not detected, cracks in the animated material appear. Obviously, their solution works well for fracturing effect of solids. Premoze et al. [PTB*03] propose a particle position reconfiguration method. In this method, particles that belong to a fixed boundary or an inlet or outlet boundary should go back to their original positions. Although the particles on the surface of free moving boundary are distributed at equal distance apart, the positions of the inside particles are arbitrarily determined. This method cannot provide sufficient particle concentration in the under-sampled regions and thus poor particle distribution still occurs. Since there is not enough information provided on how to arbitrarily determine the positions of the inside particles, this method is not repeated in this dissertation. The problem of the poor particle distribution is also realized in CFD. Chaniotis et al. [CPK02] propose a particle re-mesh method for the physical study of fluids. In this method, particles are initialized on a grid and are re-aligned back to grid locations after the particle distribution deforms. This method only applies to fluids with fixed boundaries. Later in this chapter, it is demonstrated that the under-sampled particle regions cause unrealistic splitting of fluids during stretching, which is a common behavior of non-Newtonian fluids. To deal with the poor particle distribution, the new particle re-sampling method in this dissertation consists of a down-sampling method and an up-sampling method. The down-sampling method merges two particles if they are too close to each other. The up-sampling method inserts new particles in the under-sampled regions, which are detected using Delaunay triangulation. As a result, the whole particle re-sampling method maintains a good distribution of particles. The previous particle-based non-Newtonian models [TF88; CBP05] do not demonstrate the animations of fluid stretching and thus the poor particle distribution is not a concern in these two models.

This chapter is organized as follows. The new particle dynamics method is described in Section 4.2, including the details to compute the stress tensor and the pressure. The new particle re-sampling method is described in detail in Section 4.3. Then, the animation results are presented in Section 4.4. Finally, a summary is given in Section 4.5.

4.2 Particle Dynamics

The new non-Newtonian fluid model is a particle-based fluid model. As discussed in Chapter 3, the particle motions are governed by the Lagrangian version of the Navier-Stokes (NS) equation, which is referred to as the NS equation for short in this chapter. To compute the particle motions for non-Newtonian fluids, the stress tensor is incorporated in the NS equation as follows:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla P + \nabla \cdot T + \mu \nabla^2 \mathbf{v} + \mathbf{f}$$

Equation 4-1: Lagrangian version of the NS equation with stress tensor.

where $\mathbf{v}$ is the velocity, t the time, ρ the density, P the pressure, T the stress tensor, μ the viscosity constant, and $\mathbf{f}$ the summation of any other forces such as gravity. Equation 4-1 is evaluated at each particle location under the SPH formulation. The evaluation of the viscosity term is the same as that in [MCG03] and the external force term can be computed trivially. The evaluations of the pressure and the stress tensor terms are described in details in Sections 4.2.1 and 4.2.2, respectively.

4.2.1 Pressure

In order to enforce fluid incompressibility, the pressure term is evaluated in a process to compute the Poisson equation for pressure. The process is usually called the projection method and is commonly used in many grid-based fluid models [Sta99; FF01] and in the MPS-based fluid model [PTB*03]. In this dissertation, the projection method is applied under the SPH formulation. It has two steps: the prediction step and the correction step. Since the projection method updates the particle positions and velocities, it is also called the prediction-correction integration method, as mentioned in section 3.2.

In the prediction step, temporary velocity $\mathbf{v}_i^*$ and location $\mathbf{p}_i^*$ of particle i are computed from the NS equation without the pressure term:

$$\Delta \mathbf{v}_i^t = \left(\nabla \cdot T + \mu \nabla^2 \mathbf{v} + \mathbf{f}\right)\Delta t$$

$$\mathbf{v}_i^* = \mathbf{v}_i^t + \Delta \mathbf{v}_i^t$$

$$\mathbf{p}_i{*} = \mathbf{p}_i^t + \mathbf{v}_i^* \Delta t$$

Equation 4-2: Equations to compute temporary particle position and velocity.

where $\Delta \mathbf{v}_i^t$ is the velocity change of particle i at time t, $\mathbf{v}_i^t$ and $\mathbf{p}_i^t$ the velocity and location at time t, and Δt the time step. (In the implementation, the stress tensor term must be evaluated before Equation 4-2. For better illustration, its evaluation is described later.) In this step, particles are moved not according to the pressure gradients, and the fluid incompressibility is not satisfied, that is, the temporary velocity $\mathbf{v}_i^*$ is not divergence-free. According to the mass conservation equation:

$$\frac{1}{\rho}\frac{d\rho}{dt} + \nabla \cdot \mathbf{v} = 0 ,$$

Equation 4-3: Mass conservation equation.

the divergence of $\mathbf{v}_i^*$ is proportional to the rate of the fluid density change:

$$\nabla \cdot \mathbf{v}_i^* = \frac{1}{\rho_0} \frac{\rho_0 - \rho_i^*}{\Delta t}$$

Equation 4-4: Discrete form of mass conservation equation.

where ρ_0 is the original fluid density and ρ_i^* the temporary fluid density at particle *i*.

In the correction step, the correction velocity $\Delta \mathbf{v}_i^*$ is computed from the pressure term:

$$\Delta \mathbf{v}_i^* = -\frac{1}{\rho_i^*} \nabla P_i^{t+\Delta t} \Delta t$$

Equation 4-5: Pressure term for correction velocity.

where $P_i^{t+\Delta t}$ is the pressure of particle *i* at time $t+\Delta t$, and the pressure gradient is evaluated using Equation 3-16 as in [MCG03]. The correction velocity $\Delta \mathbf{v}_i^*$ is added to the temporary velocity $\mathbf{v}_i^*$ such that the resulting velocity $\mathbf{v}_i^{t+\Delta t}$ at time $t+\Delta t$ is divergence-free, i.e.

$$\nabla \cdot (\mathbf{v}_i^* + \Delta \mathbf{v}_i^*) = \nabla \cdot \mathbf{v}_i^{t+\Delta t} = 0 .$$

Equation 4-6: Resulting velocity of being divergence-free.

By substituting Equation 4-4 and Equation 4-5 into Equation 4-6, the Poisson equation for pressure is obtained on particle *i*:

$$\frac{1}{\rho_i^*} \nabla^2 P_i^{t+\Delta t} = \frac{\rho_0 - \rho_i^*}{\rho_0 \Delta t^2} .$$

Equation 4-7: SPH-based Poisson equation for pressure.

This SPH-based Poisson equation for pressure is similar to the one under the MPS formulation [PTB*03]. The difference is that the right hand side (RHS) of Equation 4-7 is the fluid density variation while the RHS of the MPS-based Poisson equation for pressure is the variation of the particle number density. In the grid-based fluid models [Sta99; FF01], the RHS of the Poisson equation for pressure is the divergence of the velocity field. Cummins and Rudman [CR99] demonstrate that, if the velocity divergence replaces the fluid density variation on the RHS of the Poisson equation for pressure, the resulting pressures are not stable when solved under the SPH formulation. This is due to the fact that the pressures and velocities are co-located on particles in SPH.

An equation system can be constructed by applying Equation 4-7 to all the particles and the unknown variables are the particle pressures. Later in this sub-section, the pressure Laplacian in Equation 4-7 is evaluated and the equation system is described. Once the pressure $P_i^{t+\Delta t}$ is solved from the equation system, the correction velocity $\Delta\mathbf{v}_i^*$ becomes known from Equation 4-5. The velocity $\mathbf{v}_i^{t+\Delta t}$ and location $\mathbf{p}_i^{t+\Delta t}$ of particle i at time $t+\Delta t$ can be computed:

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^* + \Delta\mathbf{v}_i^*$$

$$\mathbf{p}_i^{t+\Delta t} = \mathbf{p}_i^t + \mathbf{v}_i^{t+\Delta t}\Delta t\,.$$

Equation 4-8: Equations to compute resulting particle position and velocity.

After the correction step, the new particle velocity $\mathbf{v}_i^{t+\Delta t}$ is divergence-free according to Equation 4-6, and thus the rate of the fluid density change is zero according to Equation 4-3. Finally, the fluid incompressibility is satisfied. The whole method is called projection method because the temporary velocity $\mathbf{v}_i^*$ is projected to the divergence-free velocity $\mathbf{v}_i^{t+\Delta t}$ at the correction step.

The two steps are illustrated in Figure 4-1, where the three dots denote the three positions of particle i at time t, after the prediction step (by the white dot), and at time $t+\Delta t$, respectively, and the three arrows the directions of the three velocities, $\mathbf{v}_i^*$, $\Delta\mathbf{v}_i^*$, and $\mathbf{v}_i^{t+\Delta t}$, respectively.

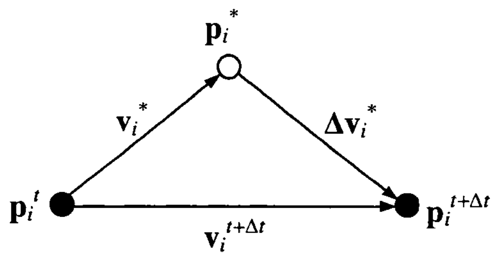


Figure 4-1: Particle movements in the projection method.

In order to construct the equation system with Equation 4-7, a new pressure Laplacian evaluation under the SPH formulation is proposed:

$$\frac{1}{\rho_i}\nabla^2 p_i = \sum_{j=1}^{n}(p_i - p_j)\frac{m_i + m_j}{2(\rho_i+\rho_j)^2}\nabla^2 W(r_{ij},h)$$

Equation 4-9: Proposed SPH-based pressure Laplacian evaluation.

where i and j are the particle indices, n the number of the neighboring particles, m the particle mass, ρ the particle density, W the kernel function, r_{ij} the distance between particle i and j, and h the smoothing length. With Equation 4-7 and Equation 4-9, the Poisson equation for pressure for particle i is:

$$p_i \sum_{j\in N(i)} M_{ij} - \sum_{j\in N(i)} p_j M_{ij} = C_i$$

$$M_{ij} = \frac{m_i + m_j}{2(\rho_i+\rho_j)^2}\nabla^2 W(r_{ij},h)$$

$$C_i = \frac{\rho_0 - \rho_i^*}{\rho_0 \Delta t^2}$$

Equation 4-10: SPH-based Poisson equation for pressure.

where i and j are the particle indices, p_i and p_j the particle pressure variables, $N(i)$ is the index set of the neighboring particles of particle i, M_{ij} the variable coefficient corresponding to particle i and j, and C_i the equation constant. Then, the equation system is:

$$\begin{bmatrix} \ddots & & & & \\ & p_i \sum_{j \in N(i)} M_{ij} & \cdots & p_k M_{ik} & \\ & \vdots & \ddots & \vdots & \\ & p_i M_{ki} & \cdots & p_k \sum_{l \in N(k)} M_{kl} & \\ & & & & \ddots \end{bmatrix} = \begin{bmatrix} \vdots \\ C_i \\ \vdots \\ C_k \\ \vdots \end{bmatrix}$$

Equation 4-11: Poisson equation system.

where k and l are also the particle indices as i and j, p_k is the particle pressure variable, $N(k)$ the index set of the neighboring particles of particle k. M_{ik}, M_{ki}, M_{kl} are the variable coefficients which are the same as M_{ij} except to be evaluated with the different particle indices. C_k is the equation constant which is the same as C_i except for particle k. If particles i and k are neighboring to each other, then M_{ik} and M_{ki} are non-zero; otherwise, zero. The system coefficient matrix is symmetric because $M_{ik} = M_{ki}$, and is positive-definite if there is at least one surface particle. (This condition can be trivially satisfied.) The pressures of the surface particles are set to be the constant air pressure, and thus the equations for the surface particles can be removed from the equation system. The system solution can be efficiently obtained using a Preconditioned Conjugate Gradient (PCG) method [Pre92].

In CFD, Shao and Lo [SL03] use a SPH-based Poisson equation for pressure to enforce fluid incompressibility. Their results are verified using only 2D experiments whereas the non-Newtonian fluid animations in this dissertation are produced in 3D. Moreover, their Laplacian evaluation is tailored for more stringent standards of instability and oscillation in CFD than those required in computer graphics. Although the proposed Laplacian evaluation is simpler than theirs, it does not cause instability or oscillation in the experiments for this dissertation.

4.2.2 Stress Tensor

As discussed in Chapter 1, the stress tensor for a non-Newtonian fluid is a non-linear function of the velocity gradient and is too complex to compute directly. A common approach to the stress tensor computation is to integrate the stress tensor rate, which is computed with the constitutive equation [GBoB04; OP02]. The constitutive equation presented in this chapter is based on a non-linear Maxwell model [EKH02]:

$$\frac{dT}{dt} = \Omega + \mu_e D' - \frac{1}{\lambda} T$$

Equation 4-12: Constitutive equation based on a non-linear Maxwell model.

where T is the stress tensor, μ_e the elasticity constant, and λ the relaxation time. Ω is called the rotational tensor and is expressed as:

$$\Omega = \frac{1}{2}(T \bullet \omega - \omega \bullet T)$$

Equation 4-13: Rotational tensor.

where ω is the fluid angular velocity and is expressed as a matrix:

$$\omega = \nabla \mathbf{v} - (\nabla \mathbf{v})^T$$

Equation 4-14: Angular velocity in matrix.

and each matrix element $\omega^{\alpha\beta}$ is

$$\omega^{\alpha\beta} = \frac{\partial \mathbf{v}^{\beta}}{\partial \mathbf{p}^{\alpha}} - \frac{\partial \mathbf{v}^{\alpha}}{\partial \mathbf{p}^{\beta}}.$$

Equation 4-15: Element of angular velocity matrix.

D' in the constitutive equation, Equation 4-12, is a traceless strain tensor:

$$D' = \frac{1}{2}D - \frac{Trace(D)}{3}I$$

Equation 4-16: Traceless strain tensor.

where D is the velocity gradient, i.e., the strain tensor, $Trace(D)$ the trace of matrix D, and I the identity matrix. D is computed as:

$$D = \nabla \mathbf{v} + (\nabla \mathbf{v})^T$$

Equation 4-17: Velocity gradient, i.e. strain tensor.

and each matrix element $D^{\alpha\beta}$ is

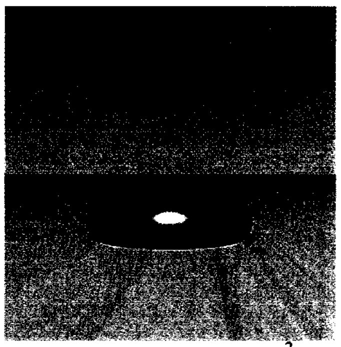
$$D^{\alpha\beta} = \frac{\partial \mathbf{v}^{\beta}}{\partial \mathbf{p}^{\alpha}} + \frac{\partial \mathbf{v}^{\alpha}}{\partial \mathbf{p}^{\beta}}.$$

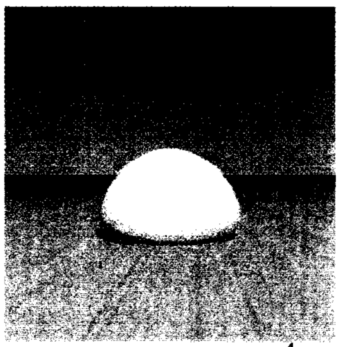
Equation 4-18: Element of strain tensor.

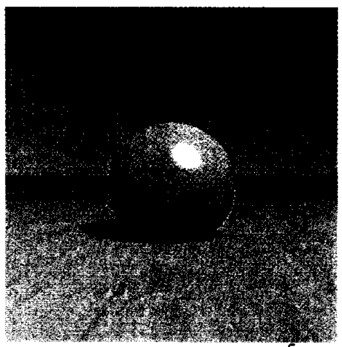
In Equation 4-15 and Equation 4-18, the Greek indices α and β denote 3D spatial coordinates. In the SPH formulation, the partial velocity derivative at position **p** is evaluated as

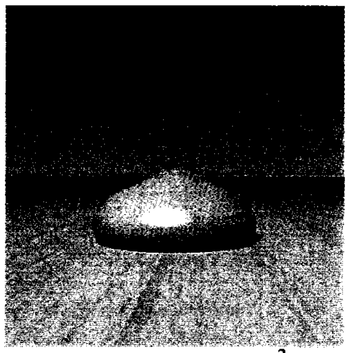
$$\frac{\partial \mathbf{v}^{\alpha}(\mathbf{p})}{\partial \mathbf{p}^{\beta}} = \sum_{j=1}^{n} \frac{m_j}{\rho_j}(\mathbf{v}_j^{\alpha} - \mathbf{v}^{\alpha}) \frac{\partial W(\mathbf{p}-\mathbf{p}_j, h)}{\partial \mathbf{p}^{\beta}}.$$

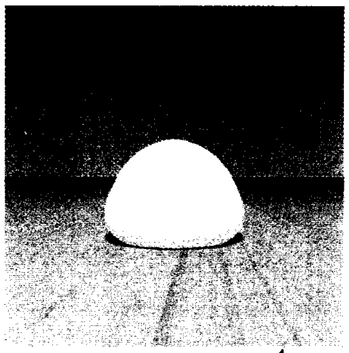
Equation 4-19: Computation of partial velocity derivative on particle.

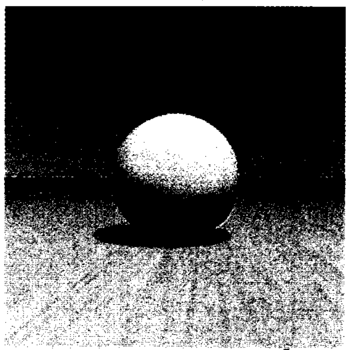

(a) $\lambda = 0.1, \mu_e = 10^3$


(b) $\lambda = 0.1, \mu_e = 10^4$


(c) $\lambda = 0.1, \mu_e = 10^5$


(d) $\lambda = 1, \mu_e = 10^3$


(e) $\lambda = 1, \mu_e = 10^4$


(f) $\lambda = 1, \mu_e = 10^5$
Bounces higher than (c)

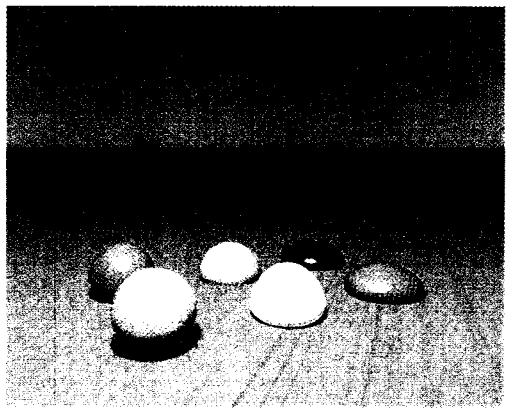

(g) Complete view of the six fluid balls.

Figure 4-2: Six fluid balls fall on the floor.

In Equation 4-12, the relaxation time λ is a characteristic constant for a non-Newtonian fluid. It characterizes the length of "memory" in which the non-Newtonian fluid has for its previous shape. $1/\lambda$ has a similar physical meaning as the material's decay rate as in the previous grid-based model [GBoB04]. Basically, the larger the λ, the more strongly a non-Newtonian fluid tries to restore to its previous shape. The elasticity constant μ_e is a factor to characterize fluid resistance to deformation. The higher the μ_e, the stronger the resistance. More information about λ and μ_e can be found in [OP02]. In Figure 4-2, six fluid balls drop onto the floor and their motions are consistent with their respective values of λ and μ_e.

The constitutive equation in the previous grid-based model [GBoB04] is similar to the one shown in here, Equation 4-12. The major difference is that their equation does not have the rotational tensor Ω which accounts for the rotational frame indifference. As a universally accepted guiding principle in material mechanics, the principle of material frame indifference states that a time-dependent material attribute must be invariant under a change of coordinate frame, that is, must be frame indifferent or frame invariant [Liu02]. In the community of physical-based fluid animation, people are familiar with frame indifference for a *vector*'s derivative, such as velocity derivative. ("Derivative" implicitly means "time derivative" in this dissertation; unless explicitly stated.) In particular, in the grid-based fluid models [Sta99], the derivative of a vector **w** is frame indifferent if it contains the advection term as follows:

$$\frac{d\mathbf{w}}{dt} = \frac{\partial \mathbf{w}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{w}$$

Equation 4-20: Frame indifferent vector derivative.

where **v** is the fluid velocity. In Equation 4-20, the left hand side is the frame indifferent derivative of vector **w**, and the second term on the right hand side is the advection term. A typical example is the frame indifferent velocity derivative that is expressed with the advection term in the grid-based fluid models:

$$\frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v}\ .$$

Equation 4-21: Frame indifferent velocity derivative.

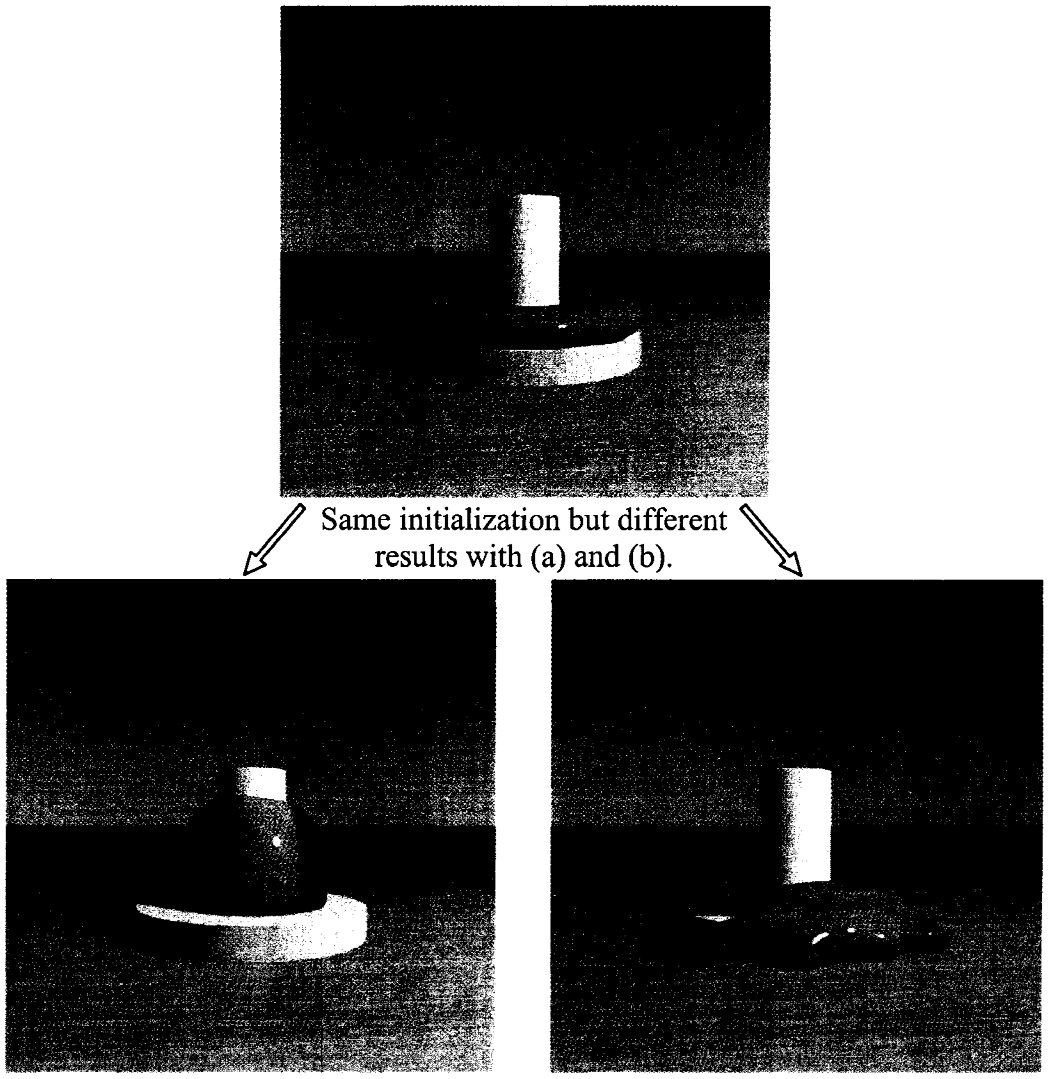
Meanwhile, in particle-based fluid models, a particle's vector derivative is measured in a frame that is always moving with the particle, thus the advection term does not appear in the vector derivative computation. In fact, the advection term in Equation 4-20 accounts for the translational frame indifference only. The fluid stress in a non-Newtonian fluid is a *tensor* attribute, and its derivative must also take into account for the rotational frame indifference [Ren02]. A well-known frame indifferent derivative of a tensor is the corotational derivative or Jaumann derivative [Liu02]. Based on this derivative, the derivative of the fluid stress tensor *T* is computed as:

$$\frac{DT}{Dt} = \frac{dT}{dt} - \Omega$$

$$\Omega = \frac{1}{2}(T \bullet \omega - \omega \bullet T)$$

Equation 4-22: Corotational derivative of fluid stress tensor.

where Ω is the coordinate transformation between the global inertial frame and a local coordinate frame which is rotating with the instantaneous fluid angular velocity ω. The constitutive Equation 4-12 is based on Equation 4-22 with Ω and thus accounts for the rotational frame indifference, whereas the constitutive equation in the previous grid-based model [GBoB04] does not without Ω. In addition to the corotational derivative, there are other frame indifferent derivatives. The discussion of them is a mathematical issue in material mechanics [Liu02] and is beyond scope of this dissertation.



(a) With rotational tensor. (b) Without rotational tensor.

Figure 4-3: Comparison of the fluid rotations with and without rotational tensor.

For visual effects in the physical-based fluid animation, the results of animating the *rod climbing* demonstrate that the presented model here with the rotational tensor Ω is more accurate in realistically animating non-Newtonian fluid phenomena than the previous grid-based model without Ω [GBoB04]. The rod-climbing is one of the most striking non-Newtonian fluid phenomena, and its description and real image can be found in [Ren00; OP02]. In this phenomenon, a rotating rod is inserted into a pool of non-

Newtonian fluid. The rotating motion creates a tension along the concentric streamlines, which leads to a force pushing the fluid inward. Consequently, the free surface rises and the fluid climbs up the rod. The similar thing happens in daily life. For example, the cooked spaghetti strands are similar to the very long and thin non-Newtonian fluid molecules. To eat spaghetti, people turn the fork and the spaghetti climbs up the fork handle, provided that the bottom of the fork is touching the plate. The rod-climbing phenomenon is animated using the presented model. One of the frames is shown in Figure 4-3(a). A comparing animation without the rotational tensor Ω is also produced, in which the rod-climbing cannot be animated. One comparing frame is shown in Figure 4-3(b).

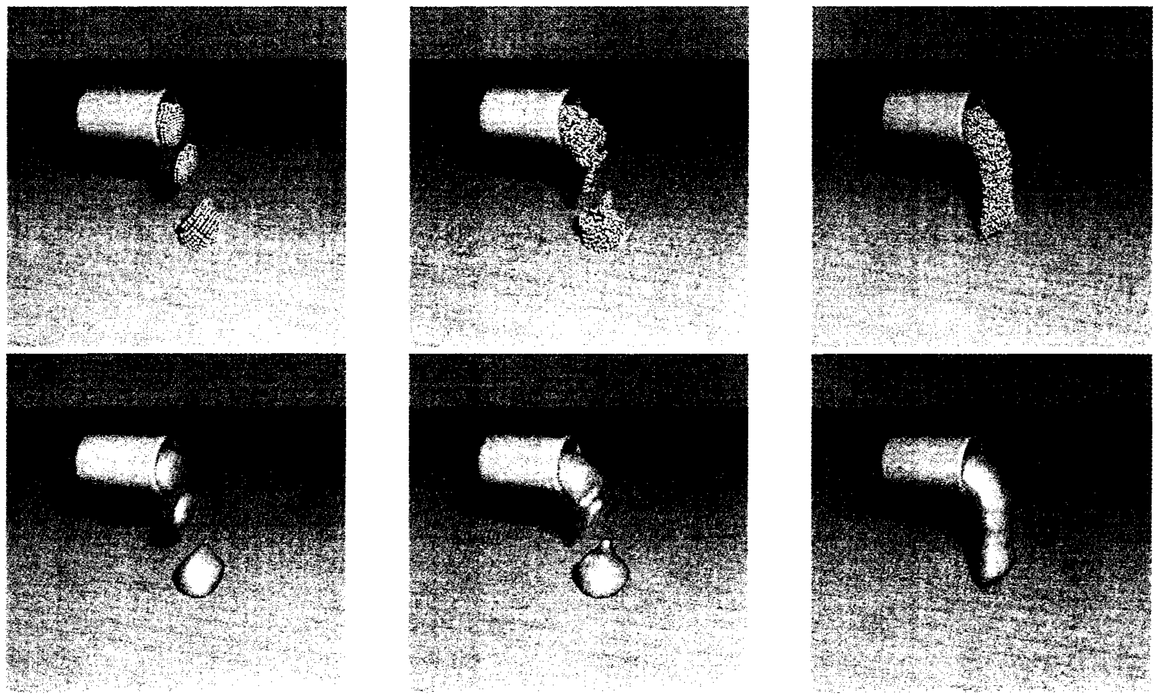
The second term D' in the constitutive equation, Equation 4-12, is the traceless strain tensor while, in [GBoB04], the matrix trace is taken out from the stress tensor T. Since the tensor-rate is continuously integrated to compute the stress tensor, using either the traceless strain tensor D' or the traceless stress tensor causes insignificant difference in the visual results. In addition, the von Mises yield condition used in [GBoB04] can be trivially embedded into Equation 4-12 by replacing T with T':

$$T' = \frac{T}{\|T\|}\max(0, \|T\| - \gamma)$$

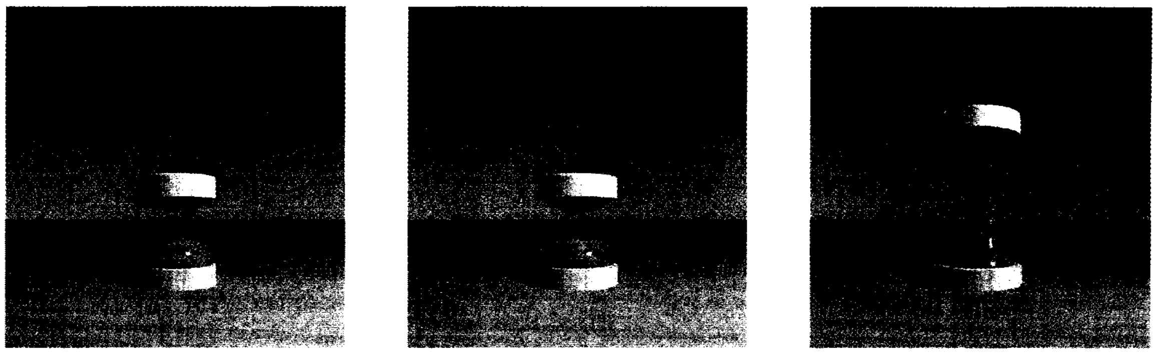
Equation 4-23: Von Mises yield condition.

where $\|T\|$ is the Frobenius norm of the stress tensor T, and γ the plastic yield value. It is noted that, if γ=0, then T=T'. For the properties of the von Mises yield condition and its impact to a non-Newtonian fluid, the detailed discussion can be found in [GBoB04].

4.3 Particle Re-sampling



(a) Unrealistic (b) Unrealistic (c) Realistic



(d) Unrealistic (e) Unrealistic (f) Realistic

The first row is the 3 particle renderings for the 3 fluid surface renderings below, respectively. The red dots in the particle rendering for (c) are the new particles inserted by the particle re-sampling. Let D_g be the cell size of the grid for particle initialization, and the jitter ratio J_a the maximum jittered distance over D_g. (a) J_a=0; (b) J_a=0.2; (c) J_a=0; (d) J_a=0; (e) J_a=0.2; (f) J_a=0.

Figure 4-4: Comparison of unrealistic and realistic fluid stretching.

As discussed at the beginning of this chapter, the poor particle distribution causes inaccurate fluid modeling. It is observed that the previous approaches work well for the animations of violent fluid motions, such as splashing and spraying. This is because the chaos in the motion appears to suppress the problem. However, these approaches do not work well for the animations of fluid stretching, which is a common motion in many non-Newtonian fluid phenomena. Figure 4-4 shows six examples of fluid stretching. In all of them, the repulsive pressure force in [DC96] is used. In [PTB*03], no detail was given on how to update particle positions randomly at each time step. To capture the idea of particle randomness, particle positions are initialized randomly in Figure 4-4(b, e). It can be seen that fluid stretching is not realistic in Figure 4-4(a, b, d, e) without particle re-sampling while it is realistic in Figure 4-4(c, f) with particle re-sampling.

The particle re-sampling method consists of a down-sampling method and an up-sampling method. Its major contribution lies in the up-sampling method. The down-sampling method is trivial. When two particles are found closer than the threshold distance ε_d, they are replaced by a new particle positioned at their center of mass. The new particle inherits their total mass and linearly interpolates their values with the weights of their masses.

The new up-sampling method is based on two well-known techniques in computational geometry: Delaunay tessellation and convex hull. The basic idea of the method is very simple, and consists of four sequential operations: (1) Detect under-sampled particle regions using Delaunay tessellation; (2) Insert a new particle into the under-sampled particle region; (3) Detect the fluid boundary using the convex hull such that the new particle is not inserted outside of the fluid; (4) Interpolate particle attributes on the new particle. The operations (1) and (2) are explained in detail in Section 4.3.1 since both are involved with Delaunay tessellation, and the operations (3) and (4) are in Sections 4.3.2 and 4.3.3, respectively.

Delaunay tessellation has been utilized to reconstruct surface from point clouds in [ABC*03]. In that application, 2D Delaunay tessellation is used to up-sample points on a 2D projection plane for the reconstructed surface. Muller et al. [MKN*04] present a point re-sampling method which applies to the object surface points only.

4.3.1 Delaunay Tessellation

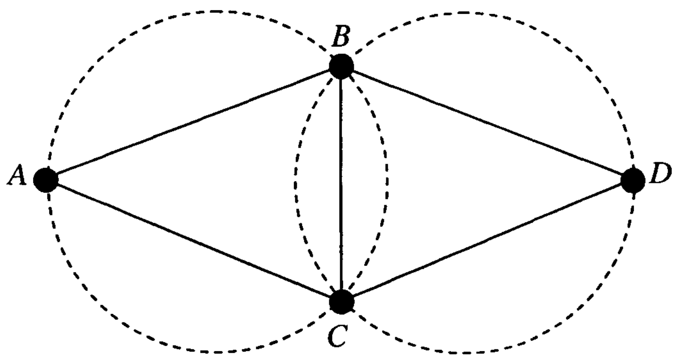


Figure 4-5: Example of Delaunay tessellation.

Delaunay tessellation is an aggregate of space-filling disjoint triangles in 2D or tetrahedra in 3D that are constructed from the given points. The most distinguished concept in Delaunay tessellation is the Delaunay condition that no given point falls inside the circum-circles of any triangles in 2D or circum-spheres of any tetrahedra in 3D. A simple illustration is shown in Figure 4-5, where two Delaunay triangles are constructed from 4 points, *A*, *B*, *C*, and *D* which are denoted by black dots. Point *A* is not inside the circum-circle of triangle *BCD* and point *D* not inside that of triangle *ABC*. The circumcircles are denoted by the dashed circles. The construction of the Delaunay tessellation has been extensively studied in computational geometry. In this dissertation, the classical Delaunay construction algorithms [Bow81; Wat81] are adopted.

Once the Delaunay tessellation is constructed from the fluid particles which are treated as simple points in the construction, the under-sampled particle regions can be easily detected according to the Delaunay condition. In particular, if the circum-sphere of a Delaunay tetrahedron is larger than the radius R_{max}, then a sparse-particle region is found because no particle is inside that circum-sphere. To up-sample in this region, a new particle is inserted at the center of the circum-sphere.

Remark: According to the Delaunay condition, this up-sampling method guarantees the minimum particle concentration that there exists at least one particle in any spherical region of radius larger than R_{max}. The smaller the R_{max}, the higher the particle concentration. R_{max} is the parameter to control the particle concentration.

The overall re-sampling method thus maintains a good distribution of particles: the threshold distance ε_d in the down-sampling method specifies the minimum particle separation while the threshold radius R_{max} the maximum empty spherical region. ε_d must be less than R_{max} such that the up-sampling operation does not result in the particle down-sampling; otherwise, there may be unnecessary system oscillation between the particle up-sampling and down-sampling. Also, the particle down-sampling takes place before the up-sampling at each time step because a particle down-sampling case may result in an up-sampling, but not vice versa. Generally, the values for ε_d and R_{max} determine the particle concentration for the animated fluid. After some trials, ε_d is chosen as $0.6{*}D_g$, and R_{max} as $0.95{*}D_g$ in this dissertation, where D_g is the initial distance between particles. These values result in sufficient particle concentration to prevent unrealistic fluid stretching but cause only small increase in particle number.

4.3.2 Convex Hull

The convex hull of a set of points is the smallest convex set that encloses the points. Usually, the convex hull is a closed series of line segments in 2D or a closed triangular mesh in 3D. A simple example is illustrated in Figure 4-6, where a convex hull is the closed series of line segments *AB*, *BC*, *CD*, *DE*, and *EA*, and encloses eight points, three inside and five on the hull, which are denoted by black dots. The convex hull construction is also extensively studied in computational geometry. For the implementation in this dissertation, the convex hull is constructed using the QuickHull algorithm in [BDH96].

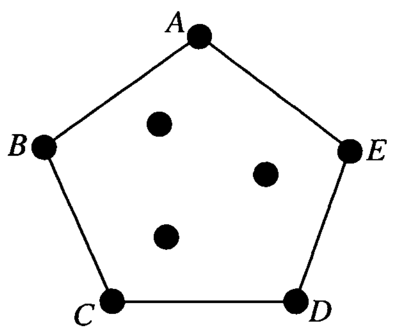
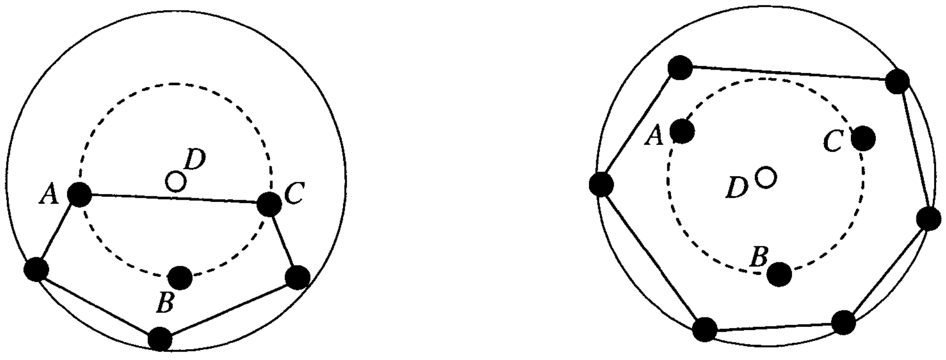


Figure 4-6: Example of convex hull.

If a new particle is inserted during the Delaunay-based up-sampling, it may fall outside of the fluid boundary. The up-sampling outside the fluid boundary must be detected and prevented; otherwise, the fluid may unrealistically expand. The detection is based on the convex hull of the neighboring particles of the new particle. If the new particle is outside the convex hull, then it is outside of the fluid boundary and is not inserted. An example is illustrated in Figure 4-7(a), where the hollow dot *D* denotes the new particle, the black dots the neighboring particles, the big circle the local neighborhood, the small dashed circle the circum-circle of 3 existing particles A, B, and C that make a Delaunay triangle, and the closed series of line segments the convex hull. In this example, particle D is outside the convex hull and thus its insertion for up-sampling is cancelled. If particle D is on or inside the convex hull as in Figure 4-7(b), then it is inserted.



(a) The new particle D is outside of the convex hull, and is not inserted.

(b) The new particle D is inside of the convex hull, and is inserted.

Figure 4-7: Detection of new particle inside or outside local fluid boundary.

The neighborhood radius for the convex hull is the same as the SPH smoothing length h, that is, two times the original particle distance. If the radius is too large, the unrealistically fluid expansion cannot be prevented. If it is too small, the new particle truly inside the fluid would not be likely inserted because the constructed convex hull would be likely too small to enclose the new particle. The current choice of the radius balances between the two extreme cases and works well in the experiments.

4.3.3 Interpolation at New Particle

When a new particle is inserted, its attributes are interpolated from its neighboring particles. Since SPH is based on the interpolation theory, the attributes can be easily computed at the new particle position using the classical SPH interpolation equation, namely, Equation 3-10. For more accurate results, the interpolated attributes are each divided by the total weight, even though the kernel is already normalized. Thus, the overall interpolation equation based on Equation 3-10 is:

$$A(\mathbf{p}) = \frac{1}{W_{total}} \sum_{j=1}^{n} A_j W_j$$

Equation 4-24: New particle interpolation equation.

where A represents the attribute to be interpolated at new particle position $\mathbf{p}$,

$$W_{total} = \sum_{j=1}^{n} W_j \text{ , and } W_j = \frac{m_j}{\rho_j} W(\mathbf{p} - \mathbf{p}_j, h) .$$

Equation 4-25: Total weight of interpolation.

In the down-sampling method, the mass is conserved since the new particle inherits the total mass of the two deleted particles. In the up-sampling method, the mass of the new particle is computed using Equation 4-24. In order to conserve mass, the mass contributed by each neighboring particle in Equation 4-24 is deducted from the corresponding particle, that is,

$$m_j = m_j - m_j \frac{W_j}{W_{total}}$$

Equation 4-26: Mass deduction from neighboring particles.

where m_j is the mass of the jth neighboring particle.

In the up-sampling method, the mass of the new particle may be very small after the computation of Equation 4-24. This usually occurs at the place where the fluid is splitting apart and the mass concentration gets lower than inside the fluid. If the new particle mass is close to zero, the particle system may become unstable. To prevent this instability, the new particle is not inserted if its mass is lower than a threshold value ε_m. In the this dissertation, $\varepsilon_m = 0.01\ m_o$, where m_o is the mass of the initialized particles.

4.4 Results

Animations in figures	N	T (s)	T_u (s)	N_s
Figure 4-2	4500	8.0	2.4	8
Figure 4-3	3100	5.7	1.6	8
Figure 4-4 (a, b, d, e)	1200	1.5	unavailable	8
Figure 4-4 (c, f)	1200	2.1	0.6	8
Figure 4-8	1100	1.3	0.6	4
Figure 4-9	1200	2.1	0.6	8

Note: N is the particle number, T the time per frame in seconds excluding the fluid surface generation time and the rendering time, T_u the up-sampling time per frame in seconds, and N_s the sub-steps per frame.

Table 4-1: Animation statistics for particle-based non-Newtonian fluid model.

The presented model is implemented within the framework of the SPH-based fluid animation as described in Chapter 3. During each time step, the particle re-sampling is performed before the particle dynamics computation. The interactions between fluids and solids are also implemented with the previous SPH-based model [MST*04]. The example animations are produced. Some select frames have been presented in the previous sections, and the others are shown in the figures at the end of this chapter. Different animations in each figure have the same initial conditions except those indicated explicitly. The animations are described in the figure captions. By default, gravity is enabled; unless indicated otherwise.

All the animations are produced using a 3.0 GHz Pentium 4 PC with 1 GB of memory running Windows XP Professional. The statistics of the animations are summarized in Table 1. Because of the high elasticity constants, each frame time step has to be divided into more sub-steps of the smaller size in order to maintain animation stability. The sub-step number is chosen by trials. For the animations with the re-sampling method, the actual particle numbers fluctuate within 20% of the listed numbers due to the particle insertions and deletions. The re-sampling method is called once per frame. About 90% of the re-sampling time is spent on the up-sampling method; in particularly, in the Delaunay construction. The fluid surface construction time and the rendering time are not included in Table 1. The former is about the same as the time per frame in Table 1 and the latter about 10 seconds per frame.

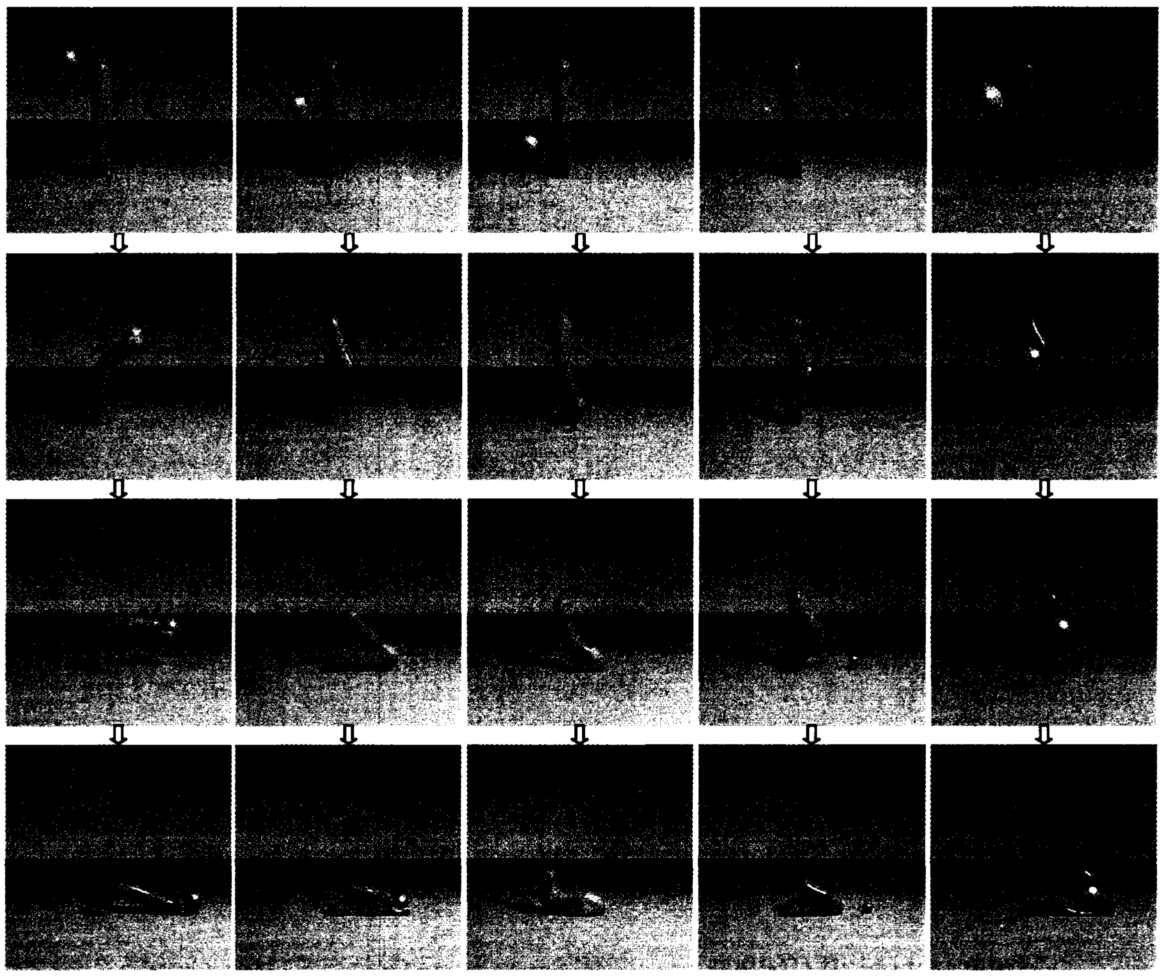
4.5 Summary

In this chapter, a new particle-based model for animating non-Newtonian fluids is presented. The animated fluids can flow, stretch, bend, bounce, split, and merge. They also can interact with solids. By changing the values of the non-Newtonian fluid parameters, i.e. elasticity constant and relaxation time, the presented model can animate a wide range of materials that fall between solids and liquids.

In terms of the fluid dynamics description, the grid-based non-Newtonian model [GBoB04] is perhaps closest to the presented model. In comparison, the presented model is more accurate for rotational fluid motions, e.g. the rod climbing in Figure 4-3. Furthermore, the presented model is able to demonstrate higher elastic behaviors, e.g. fluids being pulled up off the floor in Figure 4-9(b, c), while fluids dripping and piling in most of the example animations in [GBoB04].

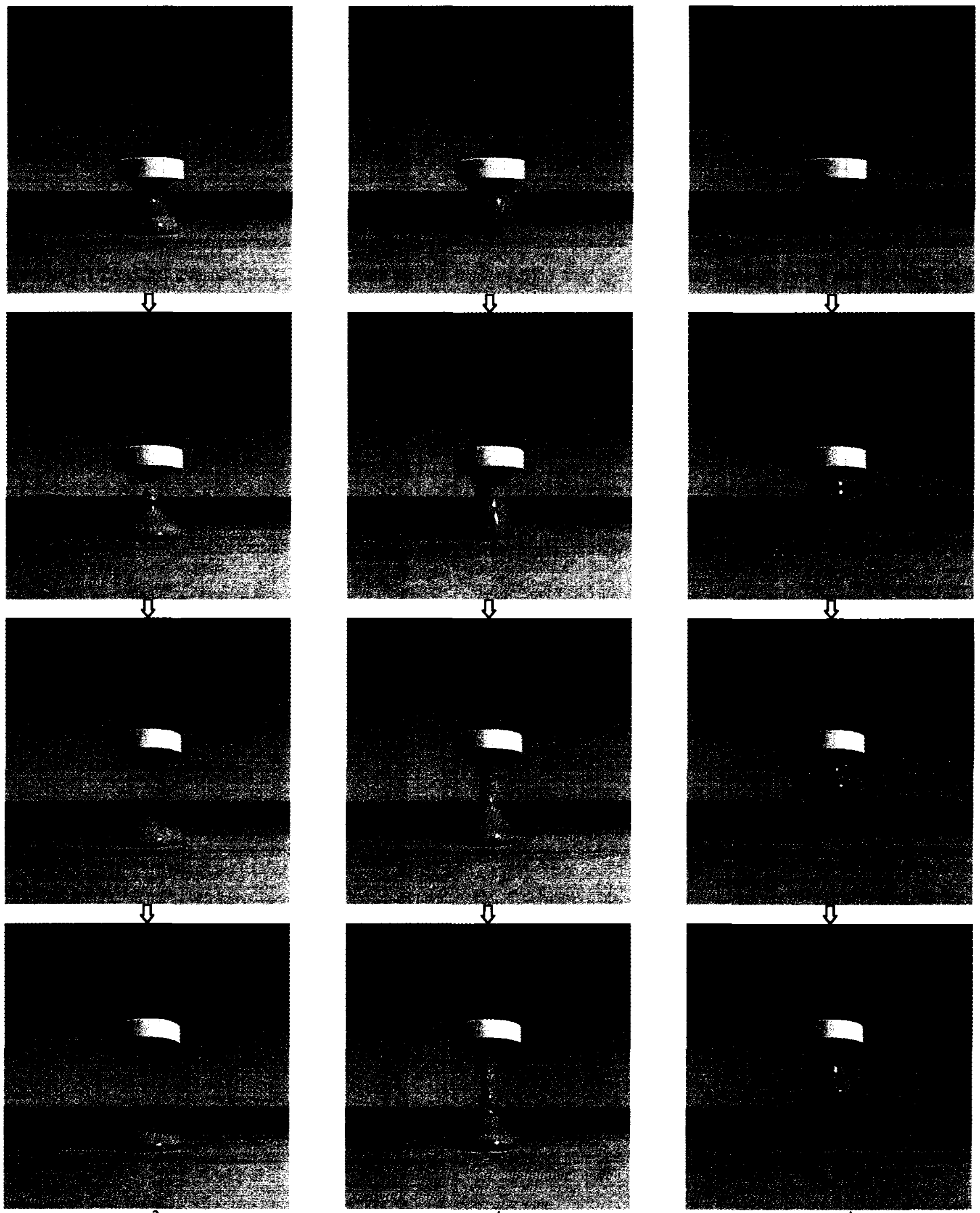
On the one hand, some fluid models have demonstrated the highly viscous fluid behaviors [CMvHT02; CBP05; GBoB04; WLK03]. In most of their example animations, the fluids are dripping, piling, and crawling on the floor, but not being pulled up off the floor as demonstrated in Figure 4-9(b, c). On the other hand, some elastic and plastic solid models have demonstrated the highly elastic or plastic behaviors [MKN*04; KAG*05; PKA*05]. In their example animations, the materials can be pulled up into air, but are not being stretched gradually such that the material neck radius is getting smaller and smaller, which is, however, demonstrated with the presented model in Figure 4-4(f) and Figure 4-9. In fact, one of these models animates material cracking under stretching [PKA*05]. Overall, in comparison with the highly viscous fluid models and the elastic and plastic solid models, the presented model seamlessly accommodates both kinds of behaviors.

The presented model does not focus on interactive animation. The example animations are produced offline even though only a few thousands of the particles are used. This is because of the time consuming up-sampling method and the sub-steps required by the high elasticity constants. Two possible efficiency improvements can be made in the near future. In the current implementation, the up-sampling method is called for every frame and the Delaunay tessellation is constructed over all the particles. It is observed that the actual up-sampling, i.e. particle insertion, takes place only in some of the frames and only at some regions of the fluid. Therefore, the animation efficiency could be significantly improved by an adaptive up-sampling method such that the up-sampling including the Delaunay construction is called when and where it is needed or most likely needed. Another possible efficiency improvement is to reduce the number of the sub-steps for each frame. The current implementation uses the explicit Euler integration, and the high elastic constant μ_e requires more sub-steps for animation stability. This limitation may be alleviated by an implicit integration scheme such that fewer or no sub-step is required.



Note: In (a, b, c), medium-sized rigid balls hit fluid towers at different positions. In (d, e), rigid balls of different sizes hit fluid towers.

Figure 4-8: Rigid balls hit fluid towers.



(a) $\mu_e=10^3$, $\lambda=0.1$.
Not pull up off the floor.

(b) $\mu_e=10^4$, $\lambda=0.1$.
Pull up off the floor a little.

(c) $\mu_e=10^4$, $\lambda=1$.
Pull up off the floor even higher than (b).

Figure 4-9: Pull up fluids of different non-Newtonian behaviors.

Chapter 5. Immiscible Fluid-Fluid Collision

5.1 Background

Fluids exhibit a wide range of motions during and after their interactions with rigid or deformable objects. As well, many fluid motions result from fluid-fluid interactions. The collision between fluids and solids has been addressed in many published papers [GHD03; CMT04; MST*04; GSLF05; CBP05]. The contribution in this chapter focuses on the collision modeling between particle-based immiscible fluids.

Keiser et al. [KMH*04] propose a contact handling model which uses a penalty force to prevent the mixing of deformable point-based objects. This model requires a two-layer representation of the animated objects, in particular, "the volumes of the objects are discretized into a set of points (or phyxels) on which external forces can be applied, and a set of surface elements (or surfels) which are animated along with the phyxels" [KMH*04]. Therefore, it would be difficult, if not impossible, for this model to work jointly with many existing particle-based fluid models [DC96; MCG03; PTB*03; CBP05], which do not represent a fluid by those two sets of points.

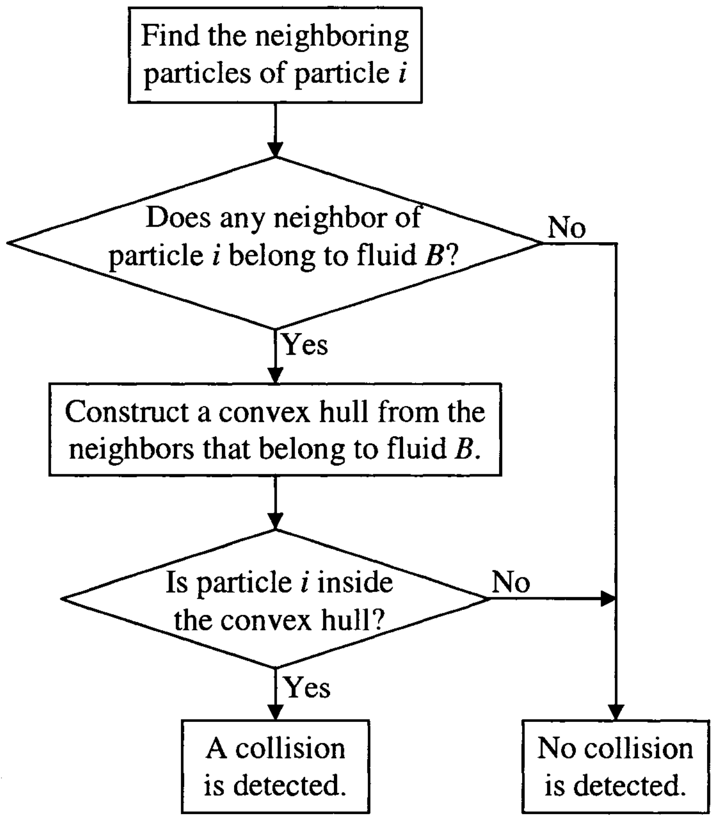
Hong and Kim [HK03] propose an interface model for bubbles in liquid. In their model, surface tension is simulated in order to maintain the interface between the bubble and the liquid. Due to the nature of the bubbling phenomenon, the air in the bubbles is always in contact with the liquid. Muller et al. [MSKG05] introduce an interface body force, which is defined along the gradient of a color field. This definition is very similar to the one for surface tension which is defined with a different color field. The interface body force acts perpendicular to the interface of the two fluids and always points from one fluid to the other fluid. Thus, such a force cannot simulate the mutual interaction in a fluid-fluid collision.

In this chapter, a new particle-based collision model for immiscible fluid animation is presented, which consists of two components: collision detection and collision response. The new model not only can prevent the mixing of fluids with each other, but also can allow one fluid to run through or to wrap around another fluid. It is noteworthy that the latter behaviors are not demonstrated in the previous fluid-fluid interaction models. In addition, the new model can easily work jointly with a typical particle-based fluid model. It requires: (1) a particle neighbor search and (2) particles with three common attributes: mass, position, and velocity. Many existing particle-based fluid models [DC96; MCG03; PTB*03; CBP05] as well as the one described in Chapter 4 can readily satisfy these two requirements. Although the particles have no explicit mass in the two previous models [PTB*03; CBP05], it is trivial to assign masses to them.

This chapter is organized as follows. Collision detection and collision response are explained in detail in Sections 5.2 and 5.3, respectively. The convex hull approximation error is discussed in Section 5.4 since the convex hull is used many times in this dissertation. Then, the animation results are presented in Section 5.5. Finally, the summary is given in Section 5.6.

5.2 Collision Detection

Collision detections of rigid and deformable solids are extensively studied in computer graphics. However, these collision detection approaches are not applicable to particle-based fluid collision. This is because rigid and deformable solids are usually modeled by fixed or flexible skeletons, geometric meshes, or regular geometries such as spheres and cubes whereas particle-based fluids are modeled by particles with no geometrical shape and no fixed structure. In this section, a new collision detection method for particle-based fluids is presented.


Find the neighboring particles of particle i
Does any neighbor of particle i belong to fluid B?
No
Yes
Construct a convex hull from the neighbors that belong to fluid B.
Is particle i inside the convex hull?
No
Yes
A collision is detected.
No collision is detected.

Note: particle i belongs to fluid A and is checked against fluid B for possible collision.

Figure 5-1: Procedure to check one particle for possible collision between two fluids.

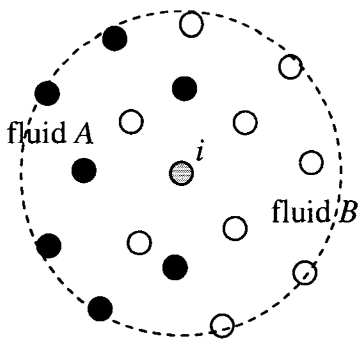
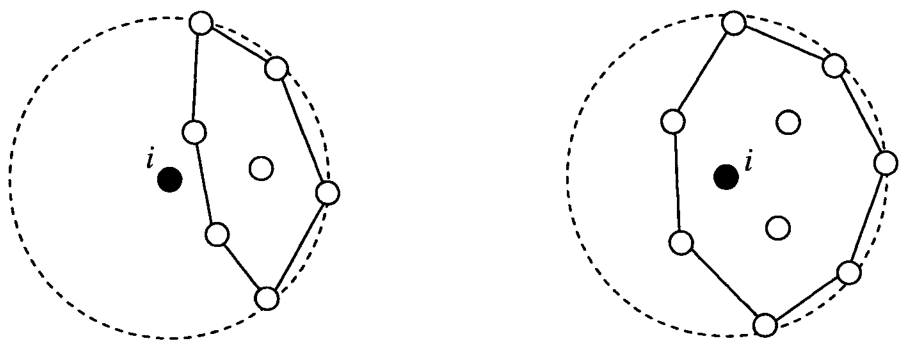


Figure 5-2: Example of particle configuration in neighborhood for collision particle.



(a) Outside convex hull (b) Inside convex hull

Figure 5-3: Use convex hull to check if a particle of one fluid is inside another fluid.

In the new method, two fluids are in collision if a particle of one fluid is found inside the other fluid. The detection procedure on the particle is illustrated in Figure 5-1. Assume that particle *i* belongs to fluid *A* and is checked against fluid *B* for possible collision between the two fluids. At first, the neighboring particles of particle *i* within the neighborhood are found by the particle neighbor search. An example of the particle configuration in the neighborhood for particle *i* in collision is illustrated in Figure 5-2 where the black and white dots denote the particles of fluid A and B, respectively, the gray dot particle *i*, the big dashed circle the neighborhood centered at particle *i*. If no neighbor of particle *i* belongs to fluid *B*, then particle *i* is definitely not inside of fluid *B* and no collision is detected on particle i; otherwise, a convex hull is constructed from the neighboring particles that belong to fluid B. If particle i is inside the convex hull, then a collision is detected; otherwise, not. The convex hulls are illustrated in Figure 5-3, where the black dots indicate particle i, the big dash circles the neighborhood of particle i, the white dots the neighboring particles of fluid B, and the segments the convex hulls. The neighbors that do not belong to fluid B are not shown in the figure for clarity. The convex hull is a local approximation of fluid B. In Figure 5-3(a), particle i is outside the convex hull and thus it is outside fluid B, whereas in Figure 5-3(b), particle i is inside the convex hull and thus inside fluid B.

In the implementation of the detection method, the convex hull is constructed with the *Quickhull* algorithm proposed by Barber et al. [BDH96]. The neighborhood radius is chosen as the SPH smoothing length, that is, two times the original particle distance. If the chosen value is too small, then the convex hull is too small to cover particle i in some cases where particle i is truly inside fluid B. If the chosen value is too large, then some particles of fluid B could be found in the neighborhood when particle i is obviously far away from fluid B. In such obvious outside case, the convex hull has to be constructed, which is not efficient. The current choice of the radius attempts to balance between accuracy and efficiency.

It is straightforward to test if a particle is inside a convex hull. In 3D, the convex hull is a closed triangular mesh. Each triangle divides the infinite space into two parts. The interior of the convex hull is contained in one part which is called the inside-part; and the other part called the outside-part. The particle is either in the inside-part or not, which can be tested out with the triangle normal. If the particle is in the inside-parts of all the triangles, then it is inside the convex hull. This test is actually equivalent to an operation in the *Quickhull* algorithm [BDH96].

In the collision detection between fluid A and B, all the particles of fluid *A* are checked against fluid *B*, and vice versa. The complexity is $O[(N_A+N_B)*D]$, where N_A is the particle number for fluid *A* and N_B for fluid *B*. *D* is the complexity of the collision detection for particle *i*, and is $O(S+C_h+T_h)$, where *S* is the complexity of the particle neighbor search, C_h of the convex hull construction, and T_h of the inside-convex-hull-test. If a grid of cells is used for the particle neighbor search as in a previous particle-based fluid model [MCG03], then $S = O(N_{gc})$ where N_{gc} is the average number of particles per grid cell. According to the *Quickhull* algorithm [BDH96], $C_h = O(N_{np}\log N_{np})$, where N_{np} is the number of the neighboring particles for particle *i*, and $T_h < C_h$.

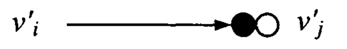
5.3 Collision Response

When a collision between two fluids is detected as in Figure 5-3(b), a particle of one fluid is mixed with the particles of the other fluid. Since the fluids are modeled by particle aggregations, particle mixture is equivalent to fluid mixture. In order to animate immiscible fluid collision, a new collision response method is presented in this section. The new method directly modifies the velocities and positions of the particles involved in the collisions. This direct modification strategy is known as geometric collision response which is already used in previous collision response models such as in the cloth animation model [VT00]. The collision response is performed once a collision is detected. The details of the modifications are explained in the following two sub-sections. The illustration is based on the collision example in Figure 5-3 (b).

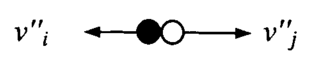
5.3.1 Particle Velocity Modification

The particle velocity modifications are based on an elastic particle-particle collision model. In the example in Figure 5-3 (b), particle *i* of fluid *A* collides with fluid *B*. It is very rare that particle *i* would collide with any particles of fluid *B*. Therefore, a virtual particle *j* of fluid *B* is created such that it collides with particle *i*. Particle *j* has the same position as particle *i*. Its mass and velocity are interpolated from the particles of fluid *B* within a spherical neighborhood. The neighborhood radius is the same as the one for particle *i*. Since all fluid particles have no physical shape, they are all represented as points. The velocities of particle i and j after the collision can be analytically solved using classical mechanics. For the convenience of implementation, the solution is described as follows.

Assume that v_i and v_j are the velocities of particle i and j, respectively. A frame of reference moving at velocity v_j is defined. (Note that it also works similarly if the frame is moving at velocity v_i.) In such a frame, the velocities of particle i and j become $v'_i = v_i - v_j$ and $v'_j = v_j - v_j = 0$, respectively, as illustrated in Figure 5-4 (a).



(a) Particles velocities before collision.



(b) Particle velocities after collision.

The black and the white dots indicate the particles, and the arrow segments the velocities.

Figure 5-4: Particle collision in moving frame.

Here, it is assumed that the collision is elastic. Thus, the kinetic energy and the momentum are conserved for the particles in the collision. The two conservation equations are:

$$m_i v''_i + m_j v''_j = m_i v'_i$$

$$\frac{1}{2} m_i v''^2_i + \frac{1}{2} m_j v''^2_j = \frac{1}{2} m_i v'^2_i$$

Equation 5-1: Conservation equations of elastic particle collision.

where m_i and v''_i are the mass and the after-collision velocity for particle i, and m_j and v''_j for particle j. Since the two particles have no shape, v''_i and v''_j are pointing along the line determined by v'_i, as illustrated in Figure 5-4 (b), and they can be expressed in terms of v'_i:

$$v''_i = a_i v'_i$$

$$v''_j = a_j v'_i \, .$$

Equation 5-2: After collision velocities in term of before-collision velocity.

With the new expressions for v''_i and v''_j, the two conservation equations become:

$$m_i = m_i a_i + m_j a_j$$

$$m_i = m_i a_i^2 + m_j a_j^2 \, .$$

Equation 5-3: Conservation equations in term of before-collision velocity.

They can be easily solved for a_i and a_j:

$$a_i = \frac{m_i - m_j}{m_i + m_j}$$

$$a_j = \frac{2m_i}{m_i + m_j}$$

Equation 5-4: Coefficients for after-collision velocities.

and, in turn, for v''_i and v''_j:

$$v''_i = \frac{m_i - m_j}{m_i + m_j} v'_i$$

$$v''_j = \frac{2m_i}{m_i + m_j} v'_i \; .$$

Equation 5-5: After-collision velocities in moving frame.

v''_i and v''_j are not the final after-collision velocities for particle i and j because they are solved in the moving frame at velocity v_j. In the inertial frame, the final after-collision velocities v'''_i and v'''_j for particle i and j are:

$$v'''_i = \frac{(m_i - m_j)v_i + 2m_j v_j}{m_i + m_j}$$

$$v'''_j = \frac{(m_j - m_i)v_j + 2m_i v_i}{m_i + m_j} \; .$$

Equation 5-6: After-collision velocities in inertial frame.

In the above collision, the momentum is conserved for particle i and j, that is,

$$\Delta M_i + \Delta M_j = 0$$

Equation 5-7: Momentum conservation in elastic collision.

where ΔM_i and ΔM_j are the momentum changes on particle i and j, respectively, and

$$\Delta M_j = m_j (v'''_j - v_j) = \frac{2m_i m_j (v_i - v_j)}{m_i + m_j} \; .$$

Equation 5-8: Momentum change for one particle.

However, the virtual particle *j* will be discarded after the collision response and thus the momentum on the whole particle system would not be conserved. To address this problem, the momentum change of particle *j* is distributed to the neighboring particles, from which particle *j*'s properties are interpolated. The weights for the distribution to the neighboring particles are the same as the weights for the interpolation. The weighted momentum change ΔM_k is added to neighboring particle *k*:

$$\Delta M_k = \frac{m_k W_k(p_j - p_k, h)}{\rho_k} \Delta M_j \, .$$

Equation 5-9: Weighted momentum change distributed to neighboring particle.

where $k = 1,\ldots, m$, and m is the number of the neighboring particles. After the distribution, the momentum is conserved for the whole particle system when the virtual particle *j* is discarded. Unfortunately, the kinetic energy of particle *j* is lost as well. This energy lost is rarely equal to the total kinetic energy change of the neighboring particles that accept the momentum distributions. Conserving the momentum and the kinetic energy at the same time is not an easy task. The above velocity modifications are only for one case of the collision response. For the whole particle system, the total kinetic energy may be decreased in some of the collision responses while increased in the others. The energy additions and deductions can compensate for each other, which suppresses the problem of the un-conserved energy. In the experiments, it is observed that the total kinetic energy is actually reduced up to 5% after all the collision responses for one time step. The energy damping is, in terms of visual effects, consistent with the damping effect in real fluid-fluid collision phenomena. Thus, it is left for future investigation to tackle the problem of the un-conserved energy.

The collisions are processed in a sequential order. The particle velocities modified in one collision case may become input data to another collision case, which makes the collision result to be dependent on the collision processing order. This is illustrated by the example in Figure 5-5, which is based on Figure 5-3. In this example, particle i and j belong to fluid *A* and they are in collision with fluid *B*. Meanwhile, particle *k* belongs to fluid *B* and is involved in both of the collisions. The velocity of particle *k* before the collisons is denoted by v_k. If the collision of particle *i* is processed first, v_k would be modified to v_{ka}. When the collision of particle *j* is processed, v_{ka} becomes the input data and would be modified to v_{kab}. If the collision processing order is reversed, that is, first for particle *j* then for particle *i*, v_k would be modified to v_{kb} and then to v_{kba}. v_{kba} is very likely not equal to v_{kab} which is the result of the different collision processing order. Therefore, the different orders of the collision processing may produce different results. However, such differences appear visually insignificant according to the experiments.

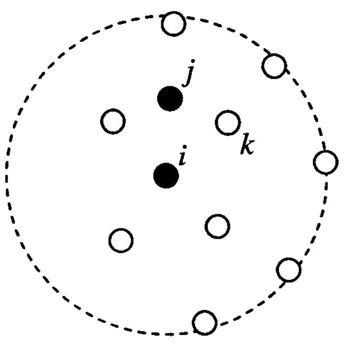


Figure 5-5: Two particles of one fluid are inside another fluid.

5.3.2 Particle Position Modification

In Figure 5-3 (b), particle *i* of fluid *A* is immersed in the particles of fluid *B*. The goal of the particle position modification is to move particle *i* towards the inside of fluid *A* and out of fluid *B*. The moving direction is the normal on the interface between the two fluids and is pointing from fluid *B* to fluid *A*.

The normal is computed at the position of particle *i*, and the computation follows the interface normal computation method in the previous SPH-based model [MSKG05]. The interface normal is similar to the surface normal described in sub-section 3.1.6. It is defined on a color field which accommodates the two fluids forming the interface, whereas the color field for the surface normal accommodates only one fluid. To compute the interface normal, the color value for the particles of one fluid is set to -0.5, and for the particles of the other fluid set to 0.5. Similar to Equation 3-30 for one fluid, the color field function $c_2(\mathbf{p})$ for two fluids is:

$$c_2(\mathbf{p}) = \sum_{j \in NA(\mathbf{p})} 0.5 \frac{m_j}{\rho_j} W(\mathbf{p} - \mathbf{p}_j, h) - \sum_{j \in NB(\mathbf{p})} 0.5 \frac{m_j}{\rho_j} W(\mathbf{p} - \mathbf{p}_j, h)$$

Equation 5-10: Color field function.

where $NA(\mathbf{p})$ and $NB(\mathbf{p})$ are the two sets of the particles of fluid A and B, respectively, within the radius h to position $\mathbf{p}$. The gradient of the color field $c_2(\mathbf{p})$ can be computed using Equation 3-11 and it is the normal of the interface between the two fluids. The gradient also can be thought of as the direction of the shortest path pointing from one fluid to another at position $\mathbf{p}$.

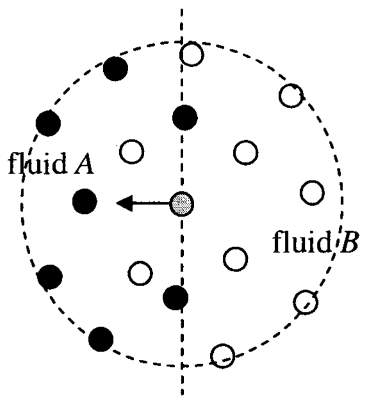


Figure 5-6: Example of particle configuration at interface.

Based on Figure 5-2, Figure 5-6 shows an example of the particle configuration at the interface, where the gray dot denotes the particle of fluid *A* at which the interface normal is computed, the big dashed circle the neighborhood centered at the gray dot, the dashed line the interface between fluid A and B, and the arrow segment the interface normal.

Based on Figure 5-3 (b), Figure 5-7 illustrates the position modification of the particle in collision. In this example, the interface normal shooting from particle *i* must intersect with the convex hull since particle *i* is inside the convex hull. As a result, particle *i* is moved to the intersection and then is not immersed in the particles of fluid *B* anymore.

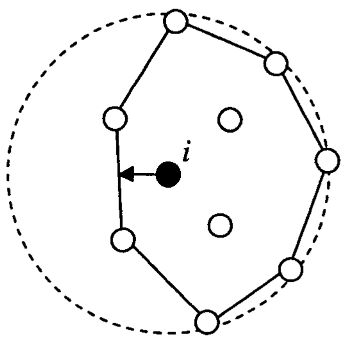


Figure 5-7: Particle in collision is moved onto enclosing convex hull.

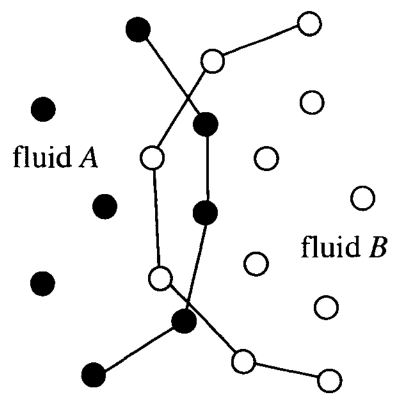
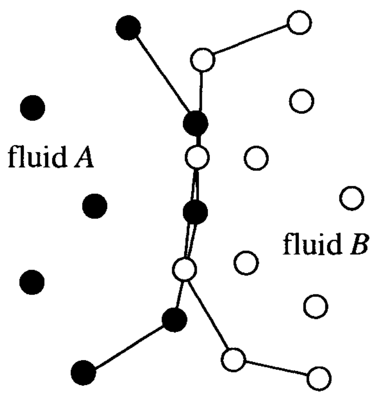
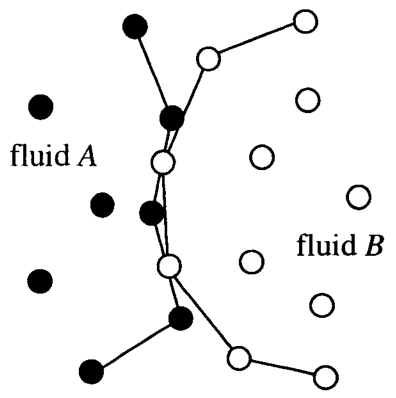


Figure 5-8: Some particles of two fluids are immersed in each other.



(a) Fluid *B* is more deformed than fluid *A* if fluid *B*'s immersed particles are first moved before *A*'s.



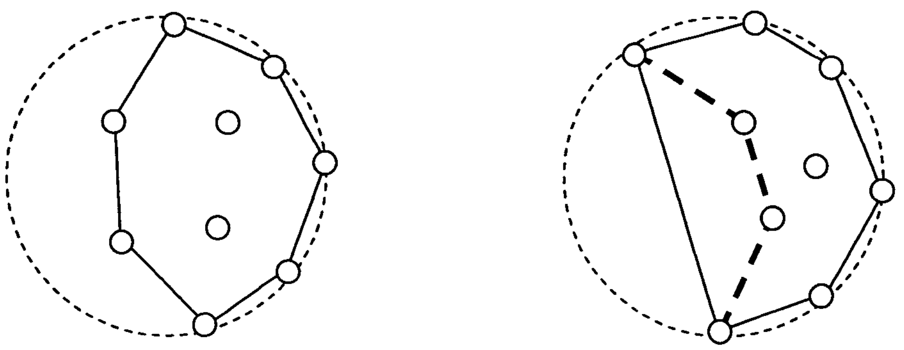
(b) Fluid *A* is more deformed than fluid *B* if fluid *A*'s immersed particles are first moved before *B*'s.

Figure 5-9: Biased fluid deformation due to the order of position modifications.

When fluid A and B are in collision, some of their particles are immersed into each other. This is illustrated in Figure 5-8 where the segments denote the fluid boundaries, and the black and white dots the particles of fluid A and B, respectively. If the immersed particles of fluid *B* are first moved out of fluid *A*, then fluid *A*'s immersed particles are less immersed than before or not immersed in fluid *B* anymore. After all the collision responses between fluid A and B are handled, fluid *B* is more deformed than fluid *A* at the collision regions. Based on Figure 5-8, an example is illustrated in Figure 5-9 (a) and a counterpart example in Figure 5-9 (b). Such a biased collision deformation on fluid A and B are not visually significant after one time step. However, if fluid *B* is always more deformed than fluid *A* at all time steps during collision, then the biased collision deformations may become quite noticeable.

An iterative process of moving immersed particles simultaneously at each time step would alleviate the biased deformation problem, but it is more computationally expensive. A cheaper approach is to move fluid *B*'s immersed particles first at odd time steps and to move fluid *A*'s first at even time steps, that is, to switch the order between fluid A and B for moving immersed particles. With this approach, both fluids have an equal chance to be the first, without the bias, to deform during collision.

5.4 Convex Hull Approximation

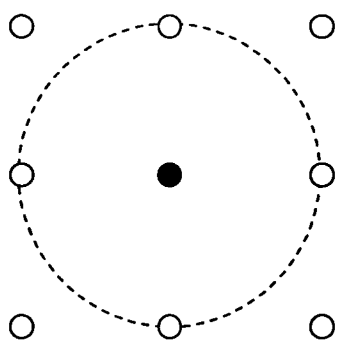


(a) Good approximation to a convex boundary.

(b} Poor approximation to a concave boundary.

Figure 5-10: Convex hull approximation to the local fluid boundary.

For the collision model in this chapter as well as for the particle re-sampling method in section 4.3, convex hull is utilized to approximate the fluid boundary in a local neighborhood and, to be clear, not to approximate the boundary of entire fluid body. If the local fluid boundary is concave, then there exists an approximation error; otherwise, the convex hull is a good fit with the local fluid boundary. These two cases are illustrated in Figure 5-10, where the big dashed circles denote the local neighborhood, the white dots the fluid particles, the series of the solid segments the convex hull, and the dashed segments in (b) the local fluid boundary. It can be seen that the convex hull is not a good fit to a local concave fluid boundary in Figure 5-10(b) whereas a good fit is shown in Figure 5-10(a).


(a) Large smoothing length with less particles.

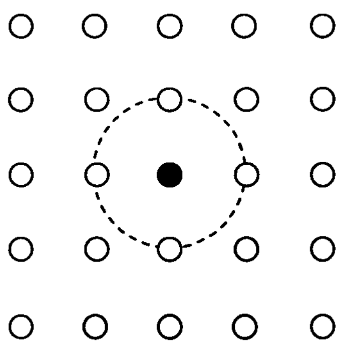

(b) Small smoothing length with more particles.

Figure 5-11: Smoothing length in proportion to particle number.

The approximation error illustrated in Figure 5-10(b) would not be a significant problem to the particle-based fluid animation. This is because the error is in proportion to the neighborhood size. In other words, the error takes place within the neighborhood and would not be bigger than the neighborhood size. As indicated before, the neighborhood size for convex hull is the same as the SPH smoothing length, that is, two times the original particle distance. As defined in sub-section 3.1.4, the SPH smoothing length is the particle interaction neighborhood size. For a certain fluid volume, the more particles are used for modeling, the smaller the original particle distance and the smaller smoothing length. This can be illustrated in Figure 5-11, where the white dots denote the fluid particles for modeling, the black dots also the fluid particles whose neighborhoods are denoted by the dashed circles. The large dashed circle means the large smoothing length whereas small circle small smoothing length. For clear illustration, the examples are shown in 2D. In (a) and (b) of Figure 5-11, the fluid particles model two square-shaped fluids of the same volume, and they are originally placed on grid locations. It can be seen that the neighborhood size, that is, the smoothing length is large with less particles in Figure 5-11 (a) whereas small with more particles in Figure 5-11 (b). Therefore, the more the particles, the smaller the convex hull approximation error. In this dissertation, a fluid is modeled by more than a thousand particles and the smoothing length is quite small. Thus, the convex hull approximation error is quite small too and would not result in significant visual artifacts.

5.5 Results

The presented fluid-fluid collision model is implemented to work with the particle-based non-Newtonian fluid model in Chapter 4. The animation results are produced on the same PC as in Chapter 4 as well. Several select frames of the animations are presented in Figure 5-12 to Figure 5-15 at the end of this chapter. Figure 5-12 to Figure 5-14 show the artificial fluid animations while Figure 5-15 some egg animations. By default, the colliding fluids in each animation have the same initial conditions except those indicated explicitly. More animation descriptions are given in the corresponding figure captions. The statistics of the animations are summarized in Table 1.

Animations in figures	N	T_c (s)	T_n (s)	N_s
Figure 5-12 (a, b)	1200	0.5	1.8	4
Figure 5-12 (c)	1800	1.0	2.6	4
Figure 5-13	1800	0.9	2.6	4
Figure 5-14	1800	0.9	2.6	4
Figure 5-15 (a)	3000	1.6	4.1	4
Figure 5-15 (b)	2500	1.3	3.3	4
Figure 5-15 (c)	3000	1.6	4.1	4

Note: N is the number of particles, T_c the average collision processing time per frame in seconds, T_n the average motion computational time per frame in seconds, excluding the fluid surface generation time and the rendering time, and N_s the sub-steps per frame.

Table 5-1: Animation statistics for particle-based fluid-fluid collision model.

The non-Newtonian fluid model in Chapter 4 has a physical parameter, elasticity constant μ_e, to control the fluid elasticity. The physical meaning of the elasticity constant is that the higher the elasticity constant, the more elastically the fluid behaves. In the animation results, the elasticity constant is varied in order to show different fluid behaviors under fluid-fluid collision. In addition, it is also demonstrated in Figure 5-14 that different collision speeds and fluid densities can cause different fluid behaviors.

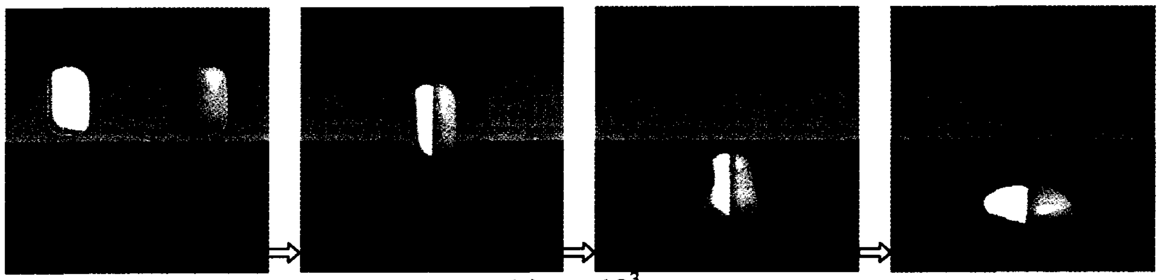
5.6 Summary

In this chapter, a new particle-based fluid-fluid collision model for immiscible fluid animation is presented. The new model simulates the fluid-fluid collision with two components: collision detection and collision response. The animation results demonstrate that the new model not only prevents the colliding fluids from mixing with each other, but also allows one fluid to wrap around or to run through another fluid. Especially, the latter behaviors are not demonstrated in the previous fluid models [HK03; KMH*04; MSKG05]. Furthermore, the new model can easily work with a generic particle-based fluid model because it requires only: (1) a particle neighbor search and (2) particles with mass, position and velocity. Many existing particle-based fluid models can trivially satisfy these two requirements.

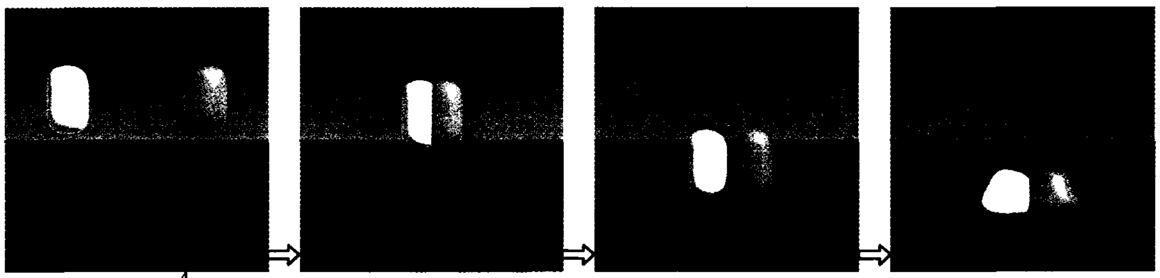
In Sections 5.2 and 5.3, the collision processing between a pair of colliding fluids is described, which can be trivially extended to handle multiple fluid collisions. As an example, assume that three fluids *A*, *B*, and *C* are colliding. The collision processing would be applied to the three pairs of the colliding fluids: *AB*, *BC*, and *AC*. An example animation of three colliding fluids is presented in Figure 5-12 (c).

The previous fluid-fluid interaction models [HK03; MSKG05] have been used to produce the animations of air bubbles in fluid and two immiscible fluids in a lava lamp. In these animations, one fluid is mainly immersed in the other fluid. In comparison, the implementation of the new model with the non-Newtonian fluid model in Chapter 4 not only animates similar immersed fluid phenomena (Figure 5-15), but also animates the collisions of fluids with free surfaces (Figure 5-12 o Figure 5-14). Thus, the new model extends the modeling scope of the particle-based fluid models.

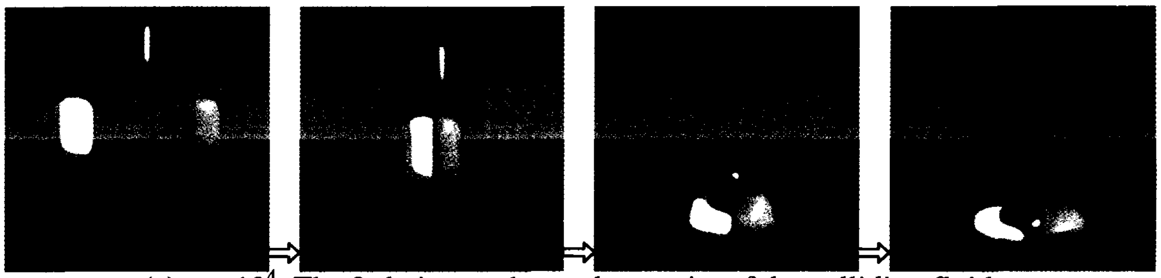
Besides immiscible fluid collision, many other fluid phenomena come from mixable fluid interaction, such as pouring milk into coffee and dripping ink into water. In the future, the animations of mixable fluids will be investigated.



(a) $\mu_e = 10^3$.

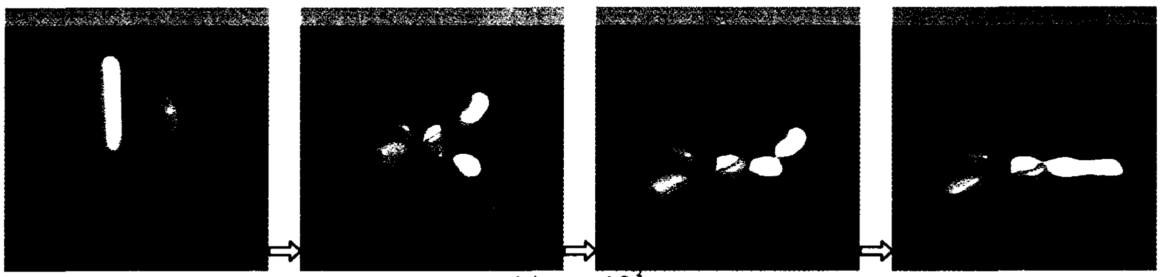


(b) $\mu_e = 10^4$. Due to the higher elasticity constant, the two fluids even bounce away.

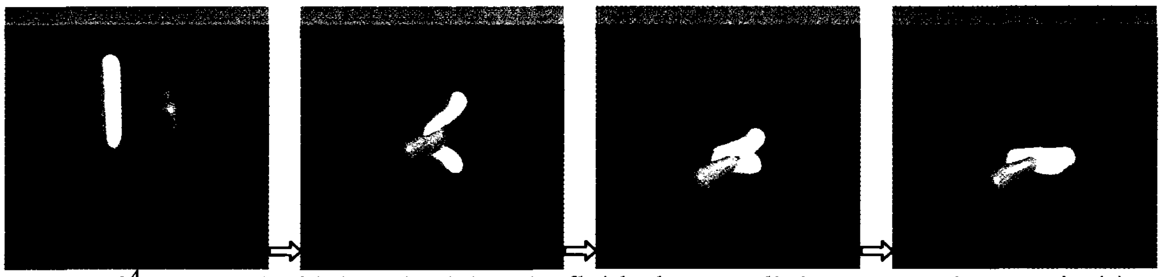


(c) $\mu_e = 10^4$. The 3rd picture shows three pairs of the colliding fluids.

Figure 5-12: Fluids are colliding with each other.

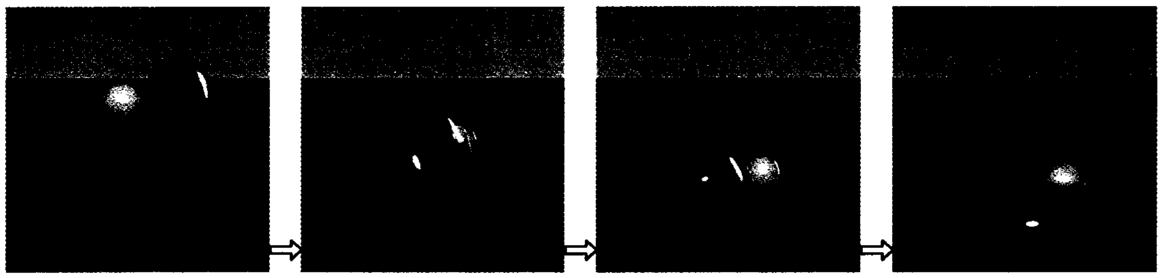


(a) $\mu_e = 10^3$.

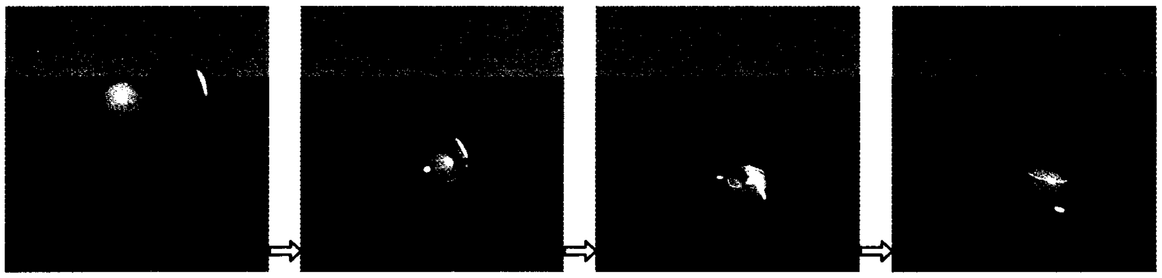


(b) $\mu_e = 10^4$. Due to the higher elasticity, the fluids do not split into many pieces as in (a).

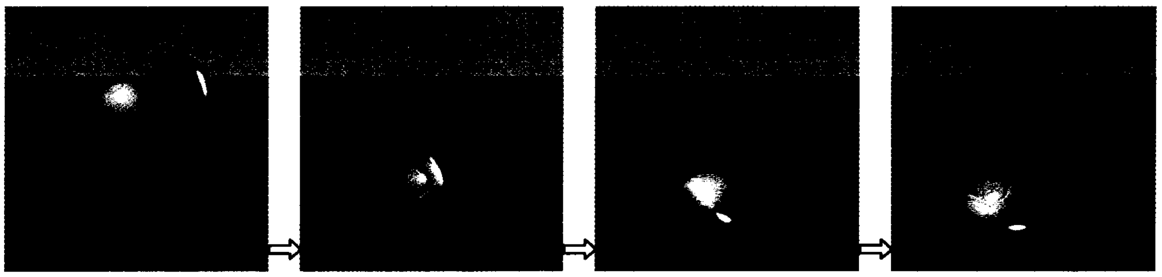
Figure 5-13: Two fluid bars are cross-colliding with each other.



(a) The colliding speed is 200. The ball runs through the disk, taking off a small piece of the disk. A hole appears on the disk after the run-through but disappears after the disk collapses on the floor.



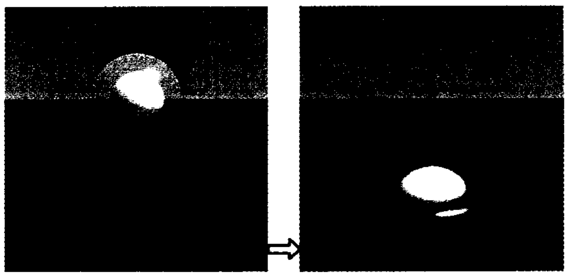
(b) The same as (a) except the colliding speed is 120. The ball cannot run through the disk due to the slower speed.



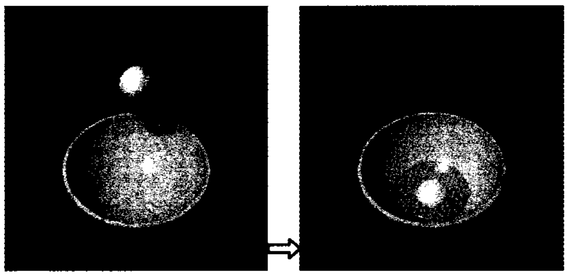
(c) The colliding speed is 200. The ball cannot run through the disk due to the reason explained in the note.

Note: $\mu_e = 10^4$ for the ball and $\mu_e = 10^3$ for the disk. The collision speed does not have physical unit because this is not a physical simulation, but it has relative physical meaning. For example, 200 is faster than 120, but both do not have physical unit such as meter per second. In (a) and (b), the particles for the ball have 10 times the mass as the particles for the disk. In (c), the particles for the ball and the disk have the same mass. The initial conditions in (c) are the same as those in (a) except the difference about the particle mass. As a result, in (c), the disk is heavy enough to stop the ball from running through.

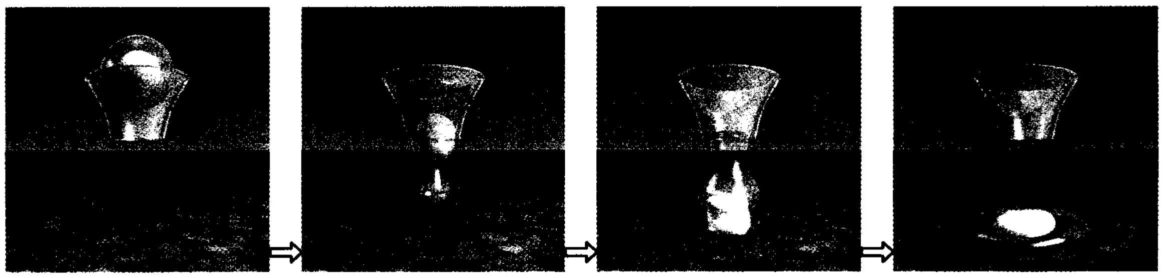
Figure 5-14: A fluid ball is running into a fluid disk.



(a) An egg falls onto the floor.



(b) An egg falls into a bowl.



(c) An egg falls through a funnel onto the floor.

Note: An egg is modeled as two fluids: egg white and yolk. Yolk is mainly immersed in egg white. μ_e = 500 for egg white and $\mu_e = 10^4$ for yolk. The particles for yolk have 10 times the mass as the particles for egg white.

Figure 5-15: Egg animations.

Chapter 6. Heating Non-Newtonian Fluids

6.1 Background

Heat is a thermal energy and its associated potential function is temperature [TPF89]. For a fluid, heat transfer occurs within the fluid and between the fluid and its environment. The heat transfer mechanisms can be classified into three broad categories: conduction, convection and radiation. This dissertation only focuses on the conductive heat transfer, which is referred to as the heat transfer for brevity. Heat convection and radiation are left for future investigation. Because of the heat transfer, temperature may vary across a fluid body. The changing temperature causes many interesting fluid phenomena, such as melting and freezing.

On the one hand, a few non-Newtonian fluid models [TF88; GBoB04; CBP05] are proposed in computer graphics. Unfortunately, these models do not simulate the heat transfer and cannot animate the heating phenomena of non-Newtonian fluids. On the other hand, the heat transfer is simulated in many other material models [TPF89; Ton91; SAC*99; CMvHT02; WLK03; MSKG05; KAG*05]. These models animate either the heating phenomena of Newtonian fluids or the material phase transitions between deformable solids and Newtonian fluids. None of these models combines heat transfer with non-Newtonian fluids.

The contribution in this chapter is a new heating model for animating non-Newtonian fluids. In terms of visual motion effects, non-Newtonian fluids are distinguished from Newtonian fluids for their strong elastic behaviors, such as bending and bouncing. The strong elastic behaviors are usually controlled by the elastic parameter, such as the elastic modulus in [GBoB04] and the spring constant in [TF88; CBP05]. In these models, the elastic parameters are constant for a fluid. It would be more interesting to vary the elastic parameter locally within a fluid body such that different parts of the fluid exhibit different elastic behaviors. For this purpose, the new heating model simulates the heat transfer in order to have variable temperatures on a fluid, and the local temperature then determines the local elastic parameter. As a result, the new heating model can animate various heating phenomena of non-Newtonian fluids, and produce interesting fluid animations that have not been demonstrated by the previous fluid models.

The new heating model is a SPH-based model. It can work jointly with the other two SPH-based fluid models presented in the last two chapters. This chapter is organized as follows. Previous heating models are discussed in Section 6.2. The heat transfer simulation is explained in detail in Section 6.3. Then, the animation results are presented in Section 6.4. Finally, a summary is given in Section 6.5.

6.2 Previous Heating Models

The heating models in computer graphics usually simulate heat transfer using the heat equation. Terzopoulos et al. [TPF89] animate the heating of deformable solids. The heat equation is solved on a hexahedral mass-spring structure which models the deformable solids. The spring stiffness varies inversely with the temperature. Tonnesen [Ton91] models solids using thermal particles. The heat equation is solved on a hexagonal configuration of the particles. The particle temperature controls how the inter-particle potential energy function behaves. "Cold" temperatures result in low potential energy minimum (that is, deeper energy well), whereas "hot" temperatures high potential energy minimum (that is, shallow energy well). Both [TPF89] and [Ton91] model fluids with free-moving particles. The interactions of the fluid particles are based on the inter-particle forces of Lennard-Jones type and are not based on the NS equation. Thus, it is not easy for these models to animate fluids with constant density, which is a common constraint in many fluid phenomena.

The SPH-based heating models are proposed in [SAC*99; MSKG05; KAG*05]. Under the SPH formulation, Stora et al. [SAC*99] solve the heat equation by two gradient computations while Muller et al. [MSKG05] and Keiser et al. [KAG*05] by a Laplacian computation. The temperature controls the fluid viscosity in [SAC*99], the fluid rest density in [MSKG05], and various material properties in [KAG*05]. It is worthy to note that [KAG*05] is able to animate the heating phenomena for elastic and plasto-elastic materials and for Newtonian fluids. In the spectrum of their modeled materials, non-Newtonian fluids are missing, which is the focus of this dissertation.

Carlson et al. [CMvHT02] propose a grid-based Newtonian fluid melting model. The model solves the heat equation with heat diffusion and heat convection on the grid. The temperature determines the fluid viscosity, which in turn controls the melting behavior.

Wei et al. [WLK03] propose a 3D cellular automata approach for animating the melting process of solid and fluid volumes. The heat transfer is based on the simple heat conduction between cells, and not based on the general heat equation. The temperature controls how much fluid a cell exchanges with its neighboring cells. In this model, the fluid motion is not based on the NS equation, and thus, the realism of fluid motion is sacrificed for the interactive animation speed.

6.3 Heat Transfer Simulation

The heat *H* in a fluid *A* can be computed by integrating over the fluid volume:

$$H = \iiint_A \rho K \theta dxdydz$$

Equation 6-1: Heat integration over fluid volume.

where ρ is the fluid density, *K* the fluid specific heat, and θ the fluid temperature. The amount of heat leaving or getting into the fluid volume per unit time is given by:

$$\iiint_A \nabla \cdot (C \nabla \theta) dxdydz$$

Equation 6-2: Heat amout leavning or getting into fluid volume per unit time.

where *C* is the heat conductivity and a constant for a fluid. A partial differential equation can be obtained by setting the heat change rate in the body, *dH/dt*, equal to Equation 6-2, and this equation is the so-called heat equation [Ton91]:

$$K \frac{\partial \theta}{\partial t} = C \frac{1}{\rho} \nabla^2 \theta .$$

Equation 6-3: Heat equation.

Equation 6-3 is used in SPH-based fluid mechanics [Mon05]. In computer graphics, the fluid specific heat *K* is assumed to be a constant for a fluid, and can be merged into the heat conductivity *C* because both of them are constants. Furthermore, for incompressible fluids, the density is assumed to be a constant as well for a fluid. Thus, a simplified heat equation is:

$$\frac{\partial \theta}{\partial t} = C \nabla^2 \theta .$$

Equation 6-4: Heat equation simplified in computer graphics.

Equation 6-4 is used in many graphical fluid models [CMvHT02; SAC*99; MSKG05; KAG*05]. To follow the tradition in computer graphics, the simplified heat equation is called the heat equation for brevity. It is noted that, in the grid-based fluid model [CMvHT02], the heat equation accommodates the advection term as follows:

$$\frac{\partial \theta}{\partial t} = C\nabla^2\theta - (\mathbf{v}\cdot\nabla)\theta$$

Equation 6-5: Heat equation in grid-based fluid model.

where **v** is the fluid velocity. However, in SPH, this advection term is not required.

In comparison to the previous heating models for a fluid, the new heating model contributes with four improvements: (1) A new SPH-based heat equation is presented, which is more flexible than the traditional heat equation; (2) The numerical integration for the heat transfer simulation is performed at smaller sub-thermal time steps within a fluid motion time step such that the heat transfer is stable even for large values of heat conductivity; (3) The heat transfer between two immiscible fluids is simulated under the SPH formulation; (4) The temperature controls the elasticity constant, an elastic property of non-Newtonian fluids. These four improvements are described in detail in the following four sub-sections.

6.3.1 SPH-Based Heat Equation

The previous SPH-based models [SAC*99; MSKG05; KAG05] simulate the heat transfer using the traditional heat equation, Equation 6-4. As indicated before, this equation is based on an assumption that the fluid density is constant. In the particle-based fluid models, it is very difficult to maintain the constant fluid density for the whole animated fluid.

The fluid density normally fluctuates [DC96] even though the projection method or the state equation of gas is used to enforce fluid incompressibility [PTB*03; MCG03]. Thus, a more flexible heat equation is proposed for the heat transfer simulation in the new heating model:

$$\frac{\partial \theta}{\partial t} = C \frac{1}{\rho} \nabla^2 \theta .$$

Equation 6-6: Heat equation incorporating fluid density.

Equation 6-6 incorporates the fluid density and is not dependent on the assumption of constant fluid density anymore.

To evaluate Equation 6-6, a new SPH-based evaluation of the temperature Laplacian on particle i is proposed:

$$\frac{1}{\rho_i} \nabla^2 \theta_i = \sum_{j=1}^{n} \frac{4m_j}{(\rho_i + \rho_j)^2} (\theta_j - \theta_i) \nabla^2 W(|r_i - r_j|, h)$$

Equation 6-7: Proposed SPH-based temperature Laplacian evaluation.

where θ_i and θ_j are the particle temperatures and the others are defined the same as in Equation 4-9, a SPH-based pressure Laplacian evaluation. The traditional SPH-based temperature Laplacian evaluation [Mon05] is stable, but the stability is confirmed only in 2D experiments for fluid mechanics. According to the experiments of 3D fluid animations for this dissertation, Equation 6-7 results in a more stable heat transfer. After the temperature Laplacian is evaluated, the particle temperature can be integrated from the rate of change of temperature according to Equation 6-6.

6.3.2 Thermal Sub-Time Step

Traditionally, the time step for integrating the temperature rate from the heat equation is the same as the time step for the fluid motion integration. It is observed that, with the traditional time step, large values of the heat conductivity C could cause overheating or oscillation of heat exchanges between particles. To address this problem, the heat transfer simulation is called iteratively within a fluid motion time step. Each call uses a thermal sub-time step, smaller than the fluid motion time step. In the implementation, the thermal sub-time step ranges from one to one eighth of the fluid motion time step. Correspondingly, the heat transfer simulation is repeated from one to eight times. The higher value of heat conductivity, the smaller the thermal sub-time step and the more iterations. This solution appears to incur a longer computational time. However, the extra time cost is insignificant. According to the experiments, the increased time is less than 5% of the total animation time.

6.3.3 Heat Transfer between Immiscible Fluids

The heat equation describes the heat transfer within a fluid. Stora et al. [SAC*99] model the heat transfer between a fluid and an exterior medium of constant temperature. Here, a new method to model the heat transfer between immiscible fluids is presented.

The new method is straightforward and is based on the immiscible fluid collision model in Chapter 5. Assume that two immiscible fluids A and B are close to each other. If particle i of fluid A has neighboring particles of fluid B, then particle i exchanges heat with those neighboring particles as if they were particles of fluid A. All the heat exchanges between fluid A's particles and fluid B's represent the heat transfer between fluid

A and B in the macroscopic view. Based on Equation 6-6, the heat equation for particle i is:

$$\frac{d\theta_i}{dt} = c_A\left(\frac{1}{\rho_i}\nabla^2\theta_i\right)_A + c_{AB}\left(\frac{1}{\rho_i}\nabla^2\theta_i\right)_B$$

Equation 6-8: Heat equation between immiscible fluids.

where c_A is the heat conductivity of fluid A and c_{AB} the heat conductivity between fluid A and B. For simplicity in the implementation, $c_{AB} = (c_A + c_B)/2$, where c_B is the heat conductivity of fluid B. On the right hand side of Equation 6-8, the first term is evaluated on particle i's neighboring particles of fluid A, and the second term on particle i's neighboring particles of fluid B. For the temperature Laplacian evaluation in Equation 6-8, the neighborhood radius is the same as that in the heat transfer within a fluid.

6.3.4 Temperature Control of Elastic Parameter

The heat transfer causes varying temperatures within a fluid. In the previous heating models, the changing temperature controls the spring stiffness [TPF89], the inter-particle energy function [Ton91], the fluid viscosity [SAC*99; CMvHT02], the fluid rest density [MSKG05], and many material properties [KAG*05]. Here, the temperature is used to control the elasticity constant μ_e, an elastic property of non-Newtonian fluids. In the non-Newtonian fluid model presented in Chapter 4, the elasticity constant is the same for all the particles of a non-Newtonian fluid. In the new heating model in this chapter, however, each particle has its own temperature and elasticity constant, which is determined by the temperature θ as follows:

$$\mu_e = 10^{a\theta+b}$$

Equation 6-9: Elasticity constant determined by temperature.

where a and b are the coefficients. If a is negative, then the higher the temperature, the less elastic the fluid. The opposite is true if a is positive. b is a constant for the convenience to tune the relation between the elasticity constant and the temperature. In this dissertation, $a = -0.01$ and $b = 8$ are used unless stated otherwise.

The temperature may also control the elasticity constant by functions other than Equation 6-9 or control other non-Newtonian fluid properties. For future research, it would be interesting to experiment with other temperature controls in addition to the one described in this sub-section.

6.4 Results

To demonstrate the modeling and animation capability of the new heating model, the animations of non-Newtonian fluids with heat transfer are produced. The heat transfer between fluids and solids is simulated with the previous method in [SAC*99]. Several select frames are presented in the figures shown at the end of this chapter. The descriptions for the animations are given in the corresponding figure captions. The different animations shown in the same figure have the same initial conditions except for those indicated explicitly. In the animations, green color with low transparency represents low temperature and weak elasticity, while red or yellow color with high transparency represents high temperature and strong elasticity. By default, the animated fluids are non-Newtonian and the heat transfer is simulated. One single heated non-Newtonian fluid with varying color can exhibit complex behaviors, including elastic motion and fluid flowing motion. For comparison, animations of Newtonian fluids or no heat transfer are also produced. The comparisons show the uniqueness in the animations of heating non-Newtonian fluids.

All the animations are produced on the same PC as in Chapter 4. The statistics of the animations are summarized in Table 1. The new heating model takes a very small fraction of *T*. More specifically, only about 2% of *T* is spent on the new heating model.

Animations in figures	N	T (s)	N_s
Figure 6-1	1000	1.9	4
Figure 6-2	1000	1.9	4
Figure 6-3	1300	2.5	8
Figure 6-4	1300	2.5	8
Figure 6-5	1800	3.5	8
Figure 6-6	1200	2.3	8
Figure 6-7	1200	2.3	8
Figure 6-8	2800	5.5	8
Figure 6-9	1800	3.5	8
Figure 6-10	3500	6.1	8

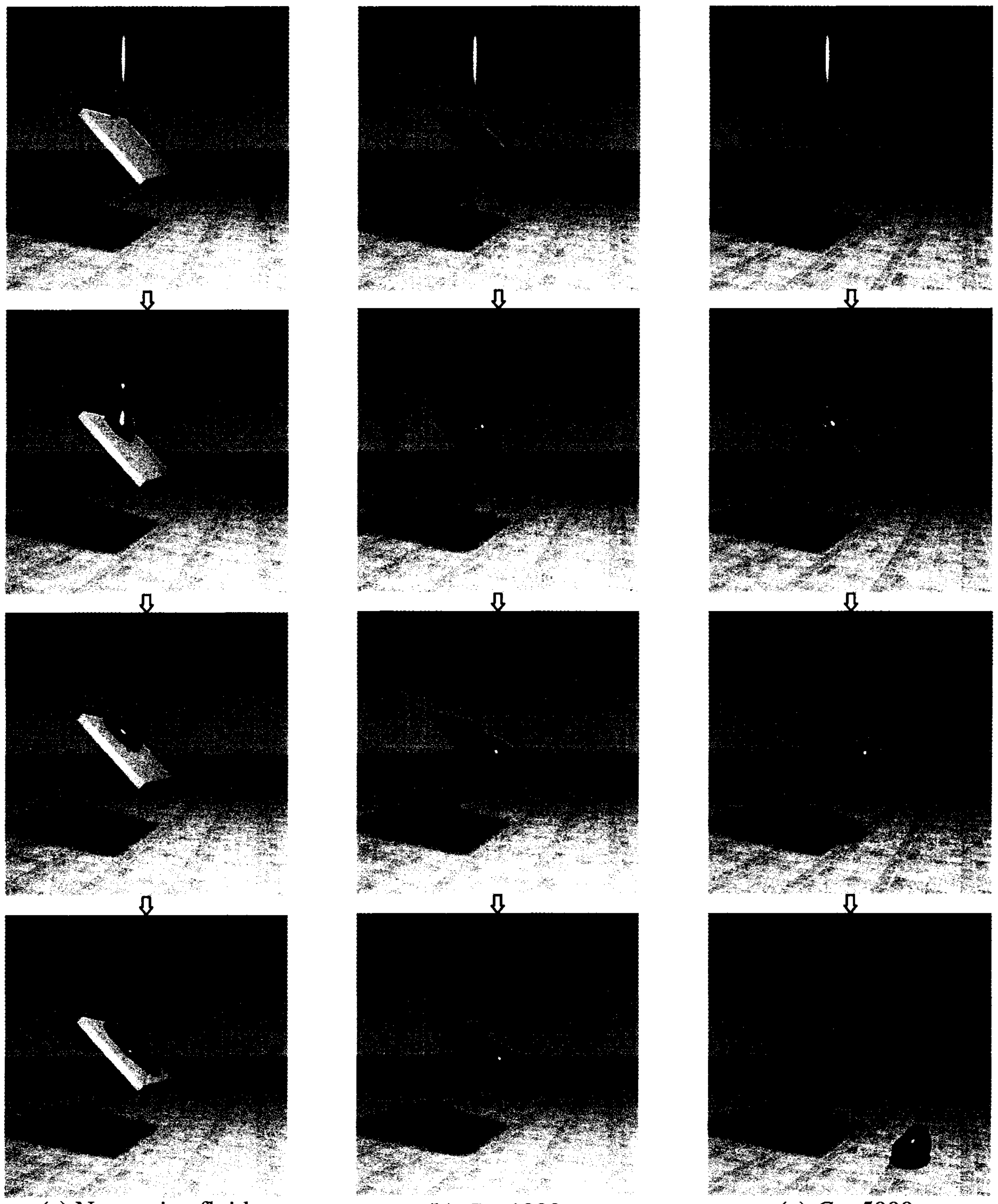
Note: N is the number of the particles in each animation, T the average motion computational time per frame in seconds, excluding the fluid surface generation time and the rendering time, and N_s the sub-steps per frame.

Table 6-1: Animation statistics for heating non-Newtonian fluids.

6.5 Summary

In this chapter, a new SPH-based heating model for non-Newtonian fluid animation is presented. In the new heating model, a new SPH-based heat equation is proposed to compute heat exchange between particles, which is more flexible than the traditional heat equation in computer graphics. By employing the thermal sub-time steps, the heat transfer simulation is stable even with large values of the heat conductivity. As an extension to the existing heat transfer methods, the new heating model simulates the heat transfer between immiscible fluids. In order to produce interesting fluid animations, temperature is used to control the elastic constant of non-Newtonian fluids. As a result, one single fluid with varying temperatures can exhibit quite complex and unique behaviors. For example, some parts of the fluid are flowing while the other parts are bending, bouncing, or resisting collapsing like solid. Such fluid phenomena are clearly demonstrated in the animation results, but not by previous fluid models.

In addition to heat conduction, convective and radiative heat transfers also occur in many heating phenomena. They could be incorporated in the future heat transfer simulation. In many heating phenomena, if a fluid changes temperature beyond a threshold value, the fluid may turn into a solid, evaporate into the air, or transform into bubbles. Such phase transitions have not been accommodated in the new heating model and will be investigated in the future.



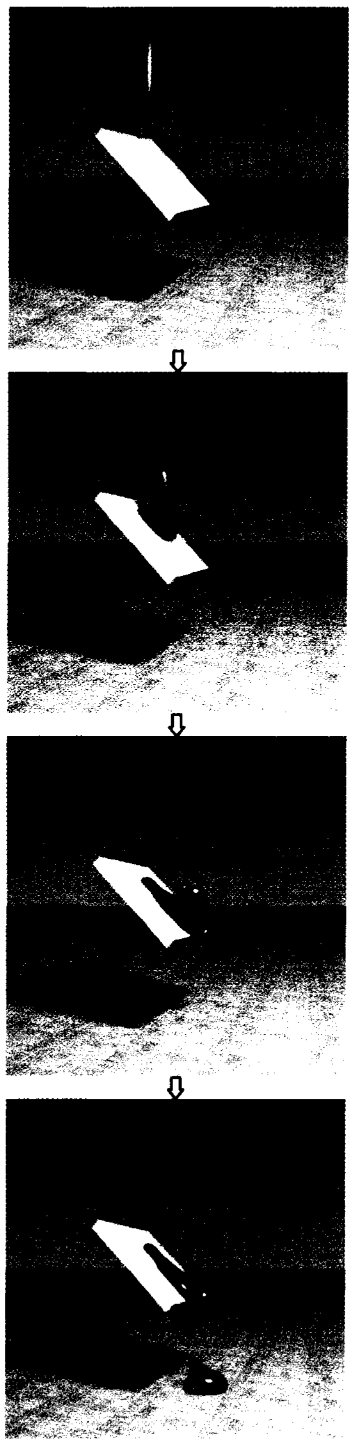
(a) Newtonian fluid, ***no*** heat transfer. Fluid bar collapsing and then flowing.

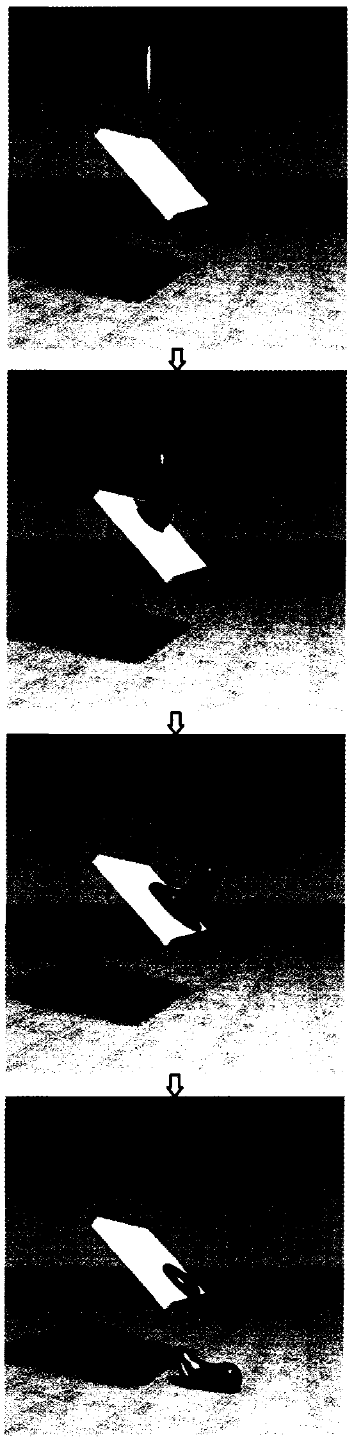
(b) $C = 1000$, ***slow*** heat transfer. Fluid bar *slowly* turning into an elastic blob.

(c) $C = 5000$, ***fast*** heat transfer. Fluid bar *quickly* turning into an elastic blob.

Note: The plate is cold and the fluid is initially hot in (b)(c).

Figure 6-1: A fluid bar heads onto a cold plate.


(a) $C = 5000$,
fast heat transfer.
Whole fluid bar melting.


(b) $C = 1000$,
slow heat transfer.
Half fluid bar melting and the other half like solid.

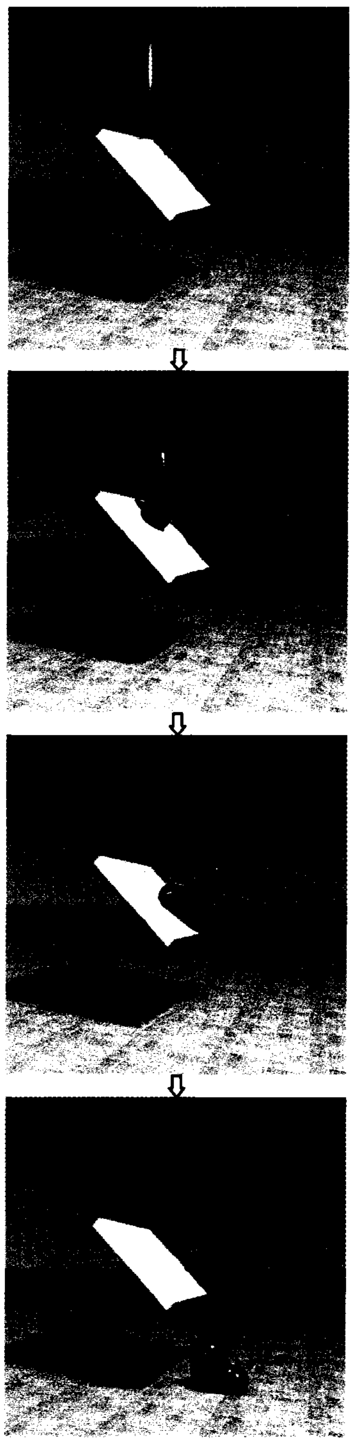
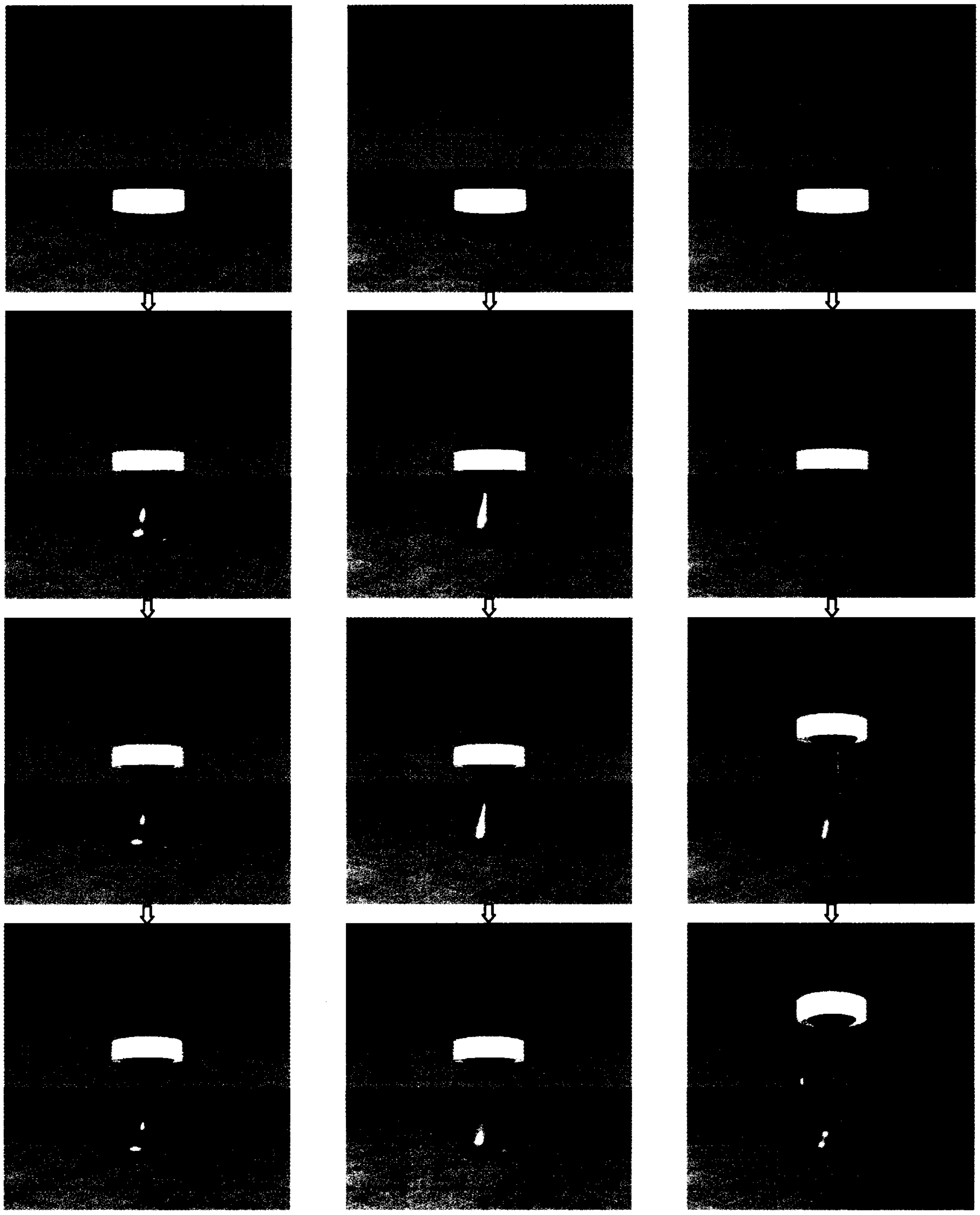

(c) $C = 200$,
very slow heat transfer.
Fluid bar melting at tip only and the rest like solid.

Figure 6-2: A cold fluid bar heads onto a hot plate.

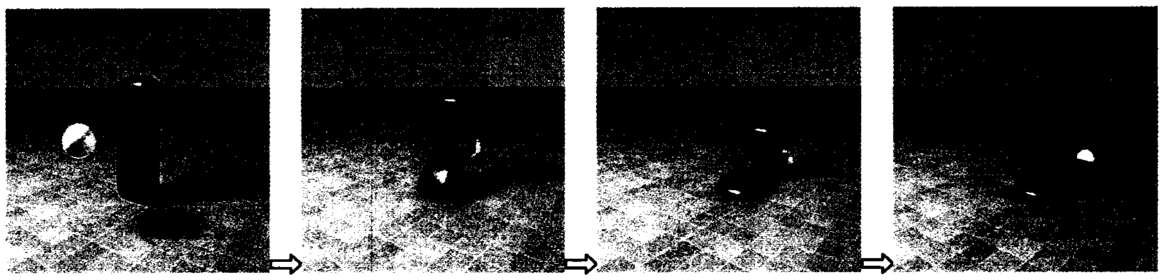


(a) $C = 8000$,
fast heat transfer.
Stretching and split.

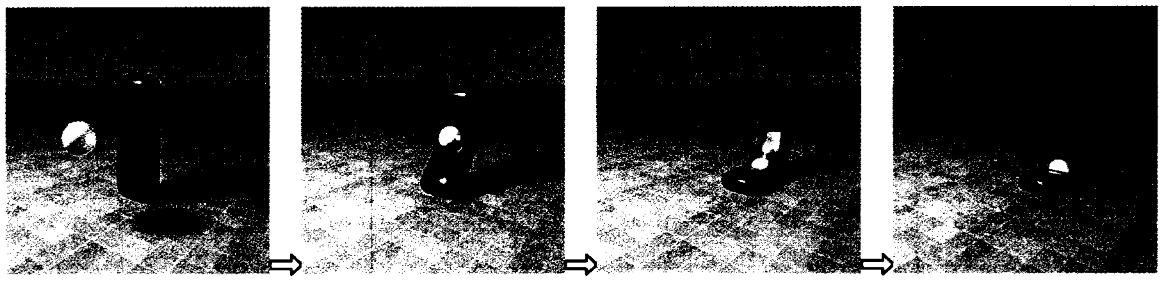
(b) $C = 500$,
slow heat transfer.
Delayed stretching and split.

(c) $C = 30$,
very slow heat transfer.
Stretching long and
bending on heated part.

Figure 6-3: A hot yellow plate pulls up a cold fluid.

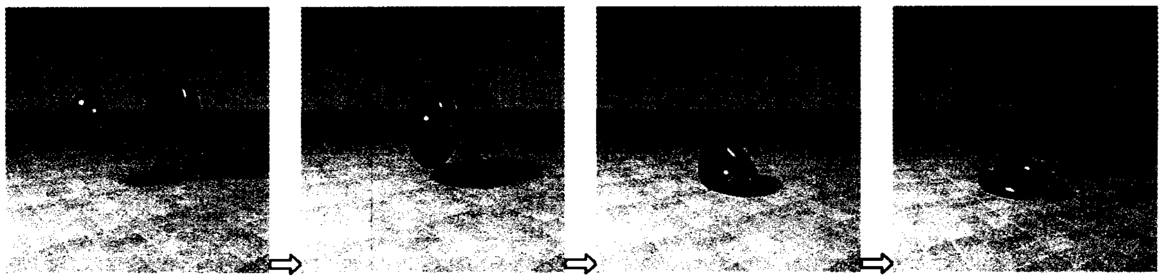


(a) C = 50000, ***very fast*** heat transfer. Ball hitting through.

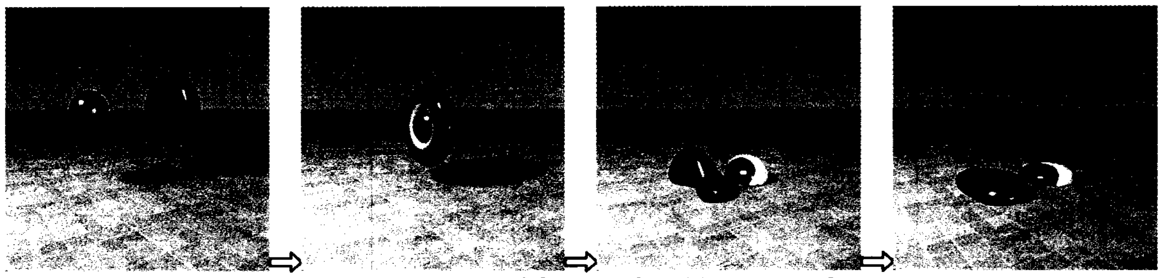


(b) C = 500, ***slow*** heat transfer. Ball not hitting through.

Figure 6-4: A hot ball hits a cold fluid.

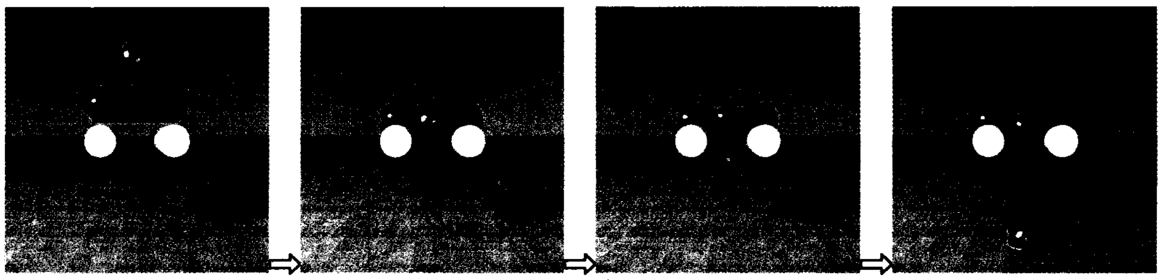


(a) No penetration ***without*** inter-fluid heat transfer.

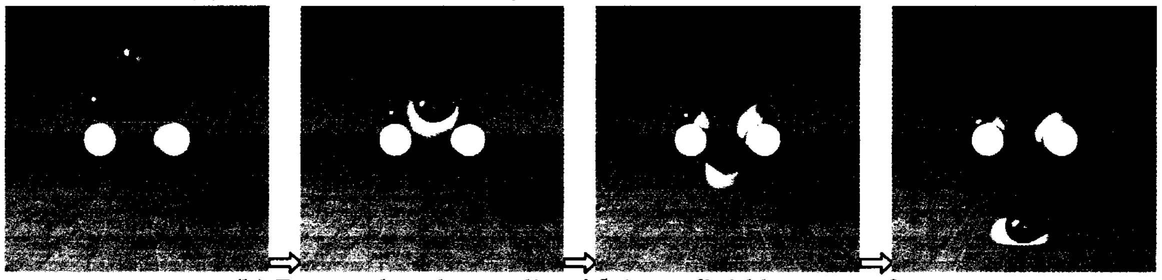


(b) Penetration ***with*** inter-fluid heat transfer.

Figure 6-5: A hot fluid ball hits a cold fluid disk.

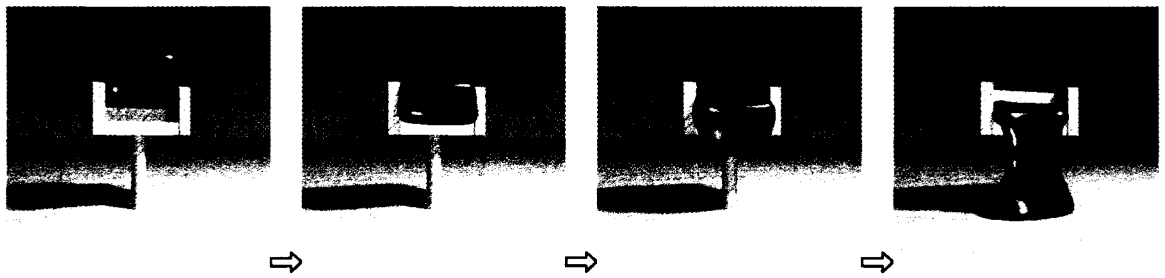


(a) Bottom bar does not split ***without*** inter-fluid heat transfer.



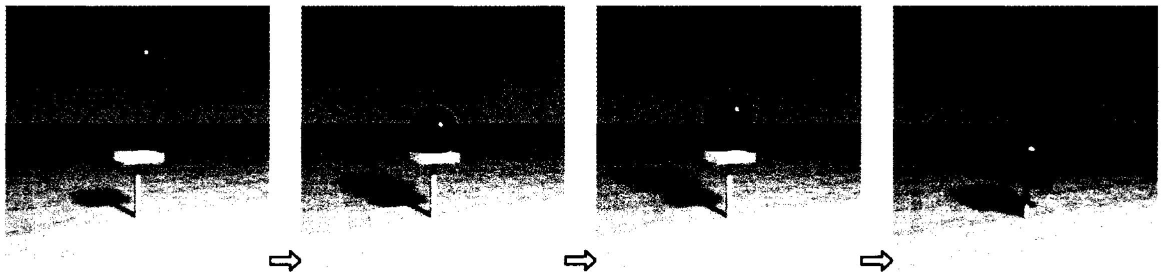
(b) Bottom bar does split ***with*** inter-fluid heat transfer.

Figure 6-6: A hot fluid bar falls onto a cold fluid bar.



The fluid box melts and forms a water-fall.

Figure 6-7: A cold fluid box falls onto a hot plate.



The ball bounces up first, and then melts down.

Figure 6-8: A cold fluid ball falls onto a hot plate.

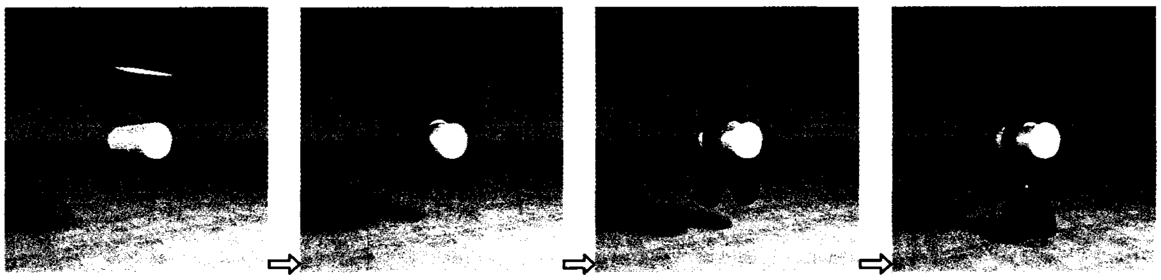

The fluid bends, stretches, and wraps around the rod. It splits at last.

Figure 6-9: A cold fluid bar falls over a hot rod.

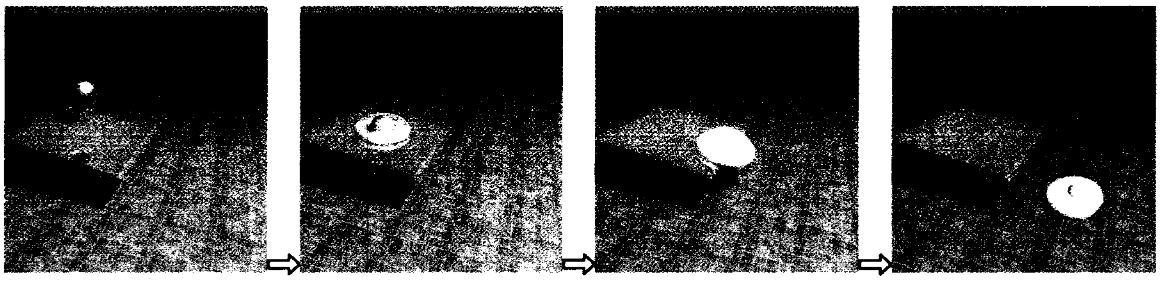

Egg white and yolk are modeled as two fluids. The egg white is transparent initially but becomes white when the egg is heated up on the plate. At last, the egg becomes elastic.

Figure 6-10: A cold egg falls onto a hot plate.

Chapter 7. Conclusion and Future Work

7.1 Conclusion

In computer graphics, many research efforts are dedicated to physical-based fluid animation. In this research area, the Navier-Stokes (NS) equation is the most comprehensive fluid dynamics description. Similar to many existing physical-based fluid models, the fluid motion computation in this dissertation is based on the NS equation.

In terms of the ways to solve the NS equation, the fluid models are divided into the grid-based models and the particle-based models. Among the particle-based fluid models, the Smoothed Particle Hydrodynamics (SPH) is a popular fluid modeling formulation. In this dissertation, fluids are modeled using particles and the fluid motions and properties are computed under the SPH formulation.

According to fluid dynamics, fluids are classified as Newtonian fluids and non-Newtonian fluids. In the physical-based fluid animation, the Newtonian fluids have drawn a lot of attentions. Many of the Newtonian fluid phenomena have been animated including waving, splashing, spraying, foams and bubbles, droplets dripping and resting on surface, as well as the fluid interactions with rigid and deformable solids. Even though Newtonian fluids and non-Newtonian fluids share many fluid properties such as surface tension and fluid incompressibility, the Newtonian fluid models are not able to simulate the stress-strain relationship for the non-Newtonian fluids and thus are not able to animate particular non-Newtonian fluid phenomena. So far, only a few attempts have been made to animate non-Newtonian fluids. This dissertation is dedicated to the animation of more non-Newtonian fluid phenomena that have not been demonstrated in the previous attempts.

In this dissertation, three original contributions are presented. They are the three fluid models for animating the non-Newtonian fluid phenomena. The first one is the particle-based non-Newtonian fluid model which is discussed in Chapter 4. This model includes a new particle-based fluid motion computation method. The new method, in turn, includes two components. The first component is the SPH-based computation for the stress-strain relationship for non-Newtonian fluids. It is distinguished from the previous fluid stress-strain computation because the presented animation demonstrates that it is more accurate for rotational fluid motions. In order to enforce the fluid incompressibility, the second component is the SPH-based projection method which is different from the MPS-based and the grid-based projection methods. The overall fluid motion computation method solves the NS equation and the constitutive equation under the SPH formulation. Based on this method, more non-Newtonian fluid phenomena can be animated. The particle-based non-Newtonian fluid model in Chapter 4 also contains a new particle resampling method. This method is able to maintain a good distribution of the particles such that the fluid stretching motion can be properly animated.

The second contribution is the particle-based immiscible fluid-fluid collision model which is presented in Chapter 5. This model contains two components for collision detection and collision response, respectively. For the collision detection, convex hull is employed to approximate the fluid boundary in the local colliding neighborhood such that the collision can be detected with a simple check of inside/outside the convex hull. For the collision response, the fluid-fluid collision is simplified as the elastic particle collision such that the after-collision velocity can be easily computed. The overall collision model has two simple requirements: (1) A particle must have mass, position, and velocity; (2) A particle neighbor search. These two requirements can be easily or trivially satisfied by many existing particle-based fluid models. In terms of the modeling capability, the presented collision model is able to separate colliding fluids and to handle multiple fluid collisions in a visually realistic way, and thus is suitable to animate immiscible fluid collsion. With this model, one fluid can penetrate or wrap around another fluid. Such phenomena are not demonstrated by the previous fluid models.

The third contribution is the heating model for non-Newtonian fluid which is presented in Chapter 6. The heating model includes four improvements. In the first improvement, a more flexible heat equation is applied for the heating simulation such that the heat transfer between the particles of variable densities is exactly conserved. The second improvement is the use of the smaller heating sub-time steps for large values of heat conductivity in order for the stable heat transfer simulation. The third improvement is a simple heat transfer method for immiscible fluids in contact. The contact detection is based on the collision detection method in Chapter 5. The last improvement is the coupling of the fluid temperature with the non-Newtonian elastic parameter such that the fluid elastic behaviors can be controlled by the temperature. With the heating model, different non-Newtonian elastic behaviors can be animated on one single fluid with variable temperatures.

The three contributions are demonstrated by the animation results shown at the end of the three corresponding chapters. They result in the animations of more non-Newtonian fluid phenomena than those animated by existing non-Newtonian fluid models.

7.2 Future Work

7.2.1 More Non-Newtonian Fluid Phenomena

Non-Newtonian fluids exhibit much diverse behaviors due to the high complexity and large variety of their microstructures. The example microstructures include suspensions such as concrete and dough, foams, and liquid crystals; granular media such as sand, gravel, and coal; and flows of materials normally regarded as solid such as flowing glaciers. So far, the most widely studied and best understood class of non-Newtonian fluids is that of polymeric fluids. The examples include molten plastics, engine oils with polymeric additives, paints, and many biological fluids such as egg white and blood [Ren00].

In CFD, there is not a super physical model that is able to cover all or most of non-Newtonian fluids. In the physical-based fluid animation, the existing fluid models including the ones in this dissertation only can animate a small fraction of non-Newtonian fluid behaviors. The techniques to animate them are far from being exhausted. It is much desired to realistically animate more non-Newtonian fluid phenomena.

7.2.2 Matter Phase Transition

Solid, fluid, and gas are three common phases of the matters. Many interesting motions of the matters can arise from phase transitions. In computer graphics, solids, fluids and gases are often individually animated. Some efforts have been made to animate solid-fluid transition [MKN*04], solids burning into gases [LIGF06], fluid-bubble transition [MSKG05]. At present, it is still a challenging topic to combine existing models of matters in different phases, or to propose a seamlessly unified model of matters, in order to animate the phase transitions among solids, fluids and gases. In terms of non-Newtonian fluids, it is also of interest to determine how the material motions result from the transitions between the non-Newtonian fluid microstructures and solid structures and from the transitions between the non-Newtonian fluid microstructures and gaseous molecules.

7.2.3 Volume Rendering

Due to the microstructure as well as the variable temperature, a non-Newtonian fluid could have variable optical attributes across its body and thus exhibit variable illumination effects. The conventional fluid rendering is based fluid surface and assumes that the whole fluid body has the same values of the optical attributes. Thus, the same deflection and color values could be applied to the entire fluid interior for the same illumination effects. This conventional fluid rendering approach cannot realistically produce the variable illumination effects inside non-Newtonian fluids.

Volume rendering is applied in the areas such as volume data visualization and animation of gaseous phenomena. This technique does not treat the material interior as uniform. However, it is a time-consuming technique in comparison to the surface rendering technique. In the future work, volume rendering could be utilized to realistically render non-Newtonian fluid interior with consideration of efficiency.

References

[BDH96] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. In *ACM Transactions on Mathematical Software*, 22(4):469-483, 1996.

[Blo97] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan-Kaufmann, July 1997.

[BK00] J. Bonet and S. Kulasegaram. A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Applied Mathematics and Computation* 126, 133-155, 2000.

[BNV96] A. Bottino, W. Nuij, and K. van Overveld. How to shrinkwrap through a critical point: an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. In *Proceedings of Implicit Surfaces '96*, pp. 53-72, 1996.

[BKZ92] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comp. Phys.*, Vol. 100, 335-354, 1992.

[CMvHT02] M. Carlson, P. J. Mucha, III R. B. Van Horn, and G. Turk. Melting and Flowing. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* 2002, 167–174.

[CMT04] M. Carlson, P. J. Mucha, and G. Turk. Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid. In *Proceedings of ACM SIGGRAPH* 2004, Los Angeles, California, August 8-12.

[CPK02] A. K. Chaniotis, D. Poulikakos and P. Koumoutsakos. Remeshed Smoothed Particle Hydrodynamics for the Simulation of Viscous and Heat Conducting Flows. *Journal of Computational Physics 182*, 67–90 (2002).

[CB00] J.K. Chen and J.E. Beraun. A generalized smoothed particle hydrodynamics method for nonlinear dynamics problems. *Comput. Methods Appl. Mech. Engrg.* 190, 225-239, 2000.

[CL95] J. X. Chen and N. D. V. Lobo. Toward Interactive-Rate Simulation of Fluids with Moving Obstacles Using Navier-Stokes Equations. *Graphical Models and Image Processing*, vol. 57, No. 2, pp. 107-116, March, 1995.

[CBP05] S.Clavet, P. Beaudoin and P. Poulin. Particle-based Viscoelastic Fluid Simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.

[CR99] S.J. Cummins and M. Rudman. An SPH projection method. *J. Comput. Phys.*,152, 584-607, 1999.

[DTG95] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation. *Proceedings of Implicit Surfaces'95* -- First Eurographics Workshop on Implicit Surfaces, pp. 171-185, Grenoble, France, 1995

[DC96] M. Desbrun and M.-P. Cani. Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies. *6th Eurographics Workshop on Animation and Simulation'96*, France.

[DC98] M. Desbrun and M.P. Cani. Active Implicit Surface for Animation. *Graphics Interface'98*. Vancouver, June, 1998.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Proceedings of SIGGRAPH'99*, CA USA, 1999.

[DC99] M. Desbrun and M.P. Cani. Space-Time Adaptive Simulation of Highly Deformable Substances. *Technical Report, INRIA*, 1999.

[Dil99] G.A. Dilts. Moving least-squares particle hydrodynamics I: consistency and stability. *Int. J. Numer. Meth. Engng.* 44: 1115-1155, 1999.

[Dil00] G.A. Dilts. Moving least-squares particle hydrodynamics II: conservation and boundaries. *Int. J. Numer. Meth. Engng.* 48: 1503-1524, 2000.

[Dur01] R. Durikovic. Animation of soap bubble dynamics, cluster formation and collision. *Computer Graphics Forum (Proceedings of Eurographics 2001)*, Vol. 20, No. 3, 67-76 (2001).

[EMF02] D. Enright, S. Marschner, and R. Fedkiw. Animation and Rendering of Complex Water Surfaces. In *Proceedings of SIGGRAPH'2002.*

[EFFM02] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *J. Comput. Phys.*, Vol. 183, Issue 1, 83-116, 20 November 2002.

[FM96] N. Foster and D. Metaxas. Realistic Animation of Liquid. *Graphical Models and Image Processing*, vol. 58, No. 5, pp. 471-483, September, 1996.

[FM00] N. Foster and D. Metaxas. Modeling Water for Computer Animation. *Communications of the ACM*, Vol. 43, No.7, July 2000.

[FF01] N. Foster and R. Fedkiw. Practical Animation of Liquids. In *Proceedings of SIGGRAPH'2001*, Los Angeles, CA USA, August 2001.

[FHP98] Patrick Fournier, Arash Habibi, and Pierre Poulin. Simulating the Flow of Liquid Droplets. *Graphics Interface'98*, (1998).

[FN80] R. Franke and G. Nielson. Smooth interpolation of large sets of scattered data. *International Journal for Numerical Methods in Engineering*, Vol. 15, pp. 1691-1704, 1980.

[FPC01] T. Frédéric, M. Philippe, and Ch. Christophe. Fast polygonization of implicit surfaces. In *Proceedings of WSCG'2001*, pages 283-290, 2001.

[GHD03] O. Genevaux, A. Habibi, and J.-M. Dischler. Simulating fluid-solid interaction. *Graphics Interface* 2003, 31-38.

[GBoB04] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien. A Method for Animating Viscoelastic Fluids. In *Proceedings of ACM SIGGRAPH* 2004, Los Angeles, California, August 8-12, 2004.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. astr. Soc.*, 181, 375-389, 1977.

[GSLF05] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling Water and Smoke to Thin Deformable and Rigid Shells. In *Proceedings of ACM SIGGRAPH* 2005, Los Angeles, California, July 2005.

[HNC02] D. Hinsinger, F. Neyret, and M.P. Cani. Interactive Animation of Ocean Waves. *Symposium on Computer Animation*, 2002.

[HK03] J.-M. Hong and C.-H. Kim. Animation of bubbles in liquid. *Computer Graphics Forum (Proceedings of Eurographics 2003)*, Vol. 22, No. 3, 2003.

[HK05] J.-M. Hong and C.-H. Kim. Discontinuous Fluids. In *Proceedings of ACM SIGGRAPH* 2005, Los Angeles, California, July 2005.

[JN92] H. M. Jaeger and S. R. Nagel. Physics of Granular States. *Science 255*, 1524, 1992.

[Kaj86] James T. Kajiya. The rendering equation. Proceedings of the 13th annual conference on Computer graphics and interactive techniques, 143-150, 1986.

[KM90] M. Kass and G. Miller. Rapid, Stable Fluid Dynamics for Computer Graphics. *Computer Graphics*, Vol. 24, No. 4, August, 1990.

[KMH*04] R. Keiser, M. Muller, B. Heidelberger, M. Teschner, and M. Gross. Contact Handling for Deformable Point-Based Objects. *Proc. Vision, Modeling, Visualization VMV'04*, Stanford, USA, pp. 315-322, Nov. 16-18, 2004.

[KBS00] L. Kobbelt, T. Bareuther and H.-P. Seidel. Multiresolution Shape Deformations for Meshes with Dynamic Vertex Connectivity. *Computer Graphics Forum (Proceedings of Eurographics'2000)*, Vol. 19, No. 3, 2000.

[KTO96] S. Koshizuka, H. Tamako, and Y. Oka. A particle method for incompressible viscous flow with fluid fragmentation. *Comput. Fluid Dynamics J.*, 29(4), 1996.

[KNO98] S. Koshizuka, A. Nobe, and Y. Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *Int. J. Numer. Meth. Fluids*, Vol. 26, 751-769, 1998.

[KVG02] Hendrik Kueck , Christian Vogelgsang , and Guenther Greiner. Simulation and Rendering of Liquid Foams. *Graphics Interface'02*, (2002).

[LP02] A. T. Layton and M. van de Panne. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, Vol. 18, No. 1, 41-53, February, 2002.

[LL03] G. R. Liu and M. B. Liu. Smoothed particle hydrodynamics: a meshfree particle method. *World Scientific*, 2003.

[Liu02] I-Shih Liu. *Continuum Mechanics*. Springer-Verlag Berlin Heidelberg New York, 2002.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface reconstruction algorithm. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, Vol. 21, No. 4, 163-169, 1987.

[LIGF06] Frank Losasso, Geoffrey Irving, Eran Guendelman, Ron Fedkiw. Melting and Burning Solids into Liquids and Gases. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 3, pp. 343-352, May/Jun, 2006.

[Luc77] L. B. Lucy. A numerical approach to testing the fission hypothesis. *A. J.*, 82, 1013-1024, 1977.

[MY06] H. Mao and Y-H. Yang: Particle-based immiscible fluid-fluid collision. *Graphics Interface 2006*, June 7-9, 2006, Quebec City, Quebec.

[MCC*99] Lee Markosian, Jonathan M. Cohen, Thomas Crulli, and John Hughes. Skin: a constructive approach to modeling free-form shapes. In *Proceedings of SIGGRAPH 99*, pages 393-400, 1999.

[MP89] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3), 305-309, 1989.

[Mon92] J. J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543-574, 1992.

[Mon94] J. J. Monaghan. Simulating free surface flows with SPH. *J. Comput. Phys.*, 110, 399-406, 1994.

[Mon05] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703-1759, 2005.

[MFZ97] J. P. Morris, P. J. Fox, and Y. Zhu. Modeling low Reynolds number incompressible flows using SPH. *Journal of Computational Physics*, 136(1), 214-226, 1997.

[Mor00] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *Int. J. Numer. Meth. Fluids*, 33, 333-353, 2000.

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-Based Fluid Simulation for Interactive Applications. *ACM SIGGRAPH Symposium on Computer Animation (SCA) 2003.*

[MKN*04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* 2004, p141-151, 2004.

[MST*04] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of Fluids with Deformable Solids. In *Proceedings of Computer Animation & Social Agents* CASA'04, Geneva, Switzerland, pp. 159-171, July 7-9, 2004.

[MSKG05] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-Based Fluid-Fluid Interaction. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation* 2005.

[OFL01] D.A. O'Brien, S. Fisher, and M. Lin. Automatic Simplification of Particle System Dynamics. *ACM SIGGRAPH Symposium on Computer Animation (SCA) 2001.*

[OH95] J.F. O'Brien and J.K. Hodgins. Dynamic Simulation of Splashing Fluids. In *Proceedings of Computer Animation'95*, Geneva, Switzerland, April 1995.

[OP02] R. G. Owens and T. N. Phillips: Computational Rheology. Imperial College Press, 2002.

[PWHS02] M.C. Potter, D.C. Wiggert, M. Hondzo, and T.I-P. Shih. *Mechanics of Fluid* (3rd edition). Pacific Grove, CA : Brooks Cole /Thompson Learning, c2002.

[PTB*03] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R.T. Whitaker. Particle-Based Simulation of Fluids. In *Proceedings of Eurographics 2003* (P. Brunet and D. Fellner), 2003.

[Pre92] W.H. Press et al. editors. *Numerical Recipes in C: The Art of Scientific Computing*. 2nd ed., Cambridge, New York, Cambridge University Press, 1992.

[Ree83] W.T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2(2)*, pages 91-108, 1983.

[Ren00] M. Renardy: *Mathematical Analysis of Viscoelastic Flows*. Philadelphia: Society for Industrial and Applied Mathematics, c2000.

[Rob95] R. Robert. Lava Lamp. *The Mathematica Journal*, Vol. 5, Issue 4, pages 12-16, 1995.

[Roy95] T. M. Roy. Physically-Based Fluid Modeling using Smoothed Particle Hydrodynamics. Master Thesis, University of Illinois at Chicago, Chicago, Illinois, 1995.

[Sch99] B. Schlatter. A Pedagogical Tool Using Smooth Particle Hydrodynamics to Model Fluid Flow Past a System of Cylinders. Dual MS Project, Oregon State University, June 11th, 1999.

[SL03] S. Shao and E.Y.M. Lo. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, article in press, 2003.

[SF95] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. *Computer Graphics, 29 (Annual Conference Series)*, 129-136, 1995.

[Sta99] J. Stam. Stable Fluids. In *Proceedings of SIGGRAPH'99*, Los Angeles, CA USA, 1999.

[SH97] B. Stander and J.C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of SIGGRAPH 97*, pages 279-286, Aug. 1997.

[SAC*99] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel. Animating Lava Flows. *Graphics Interface'99*, Kingston, Canada, June, 1999.

[TFK*03] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Uek. Realistic animation of fluid with splash and foam. *Computer Graphics Forum (Proceedings of Eurographics 2003)*, Vol. 22, No. 3, 2003.

[TPF89] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (From goop to glop). *Graphics Interface'89*, pp.219-226, June, 1989.

[TF88] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proceedings of ACM SIGGRAPH 88*, 269–278.

[TTD00] S. Torquato, T. M. Truskett, and P. G. Debenedetti. Is Random Close Packing of Spheres Well Defined? *Phys. Rev. Lett.*, 84, 2064-2067, 2000.

[TKY02] R. Tong, K. Kaneda, and H. Yamashita. A volume-preserving approach for modeling and animating water flows generated by meatballs. *The Visual Computer*, Vol. 18, No. 8, 469 - 480, December, 2002.

[VB99] C.W.A.M. van Overveld and B. C. van den Broek. Using the implicit surface paradigm for smooth animation of triangle meshes. In *Proceedings of 1999 Computer Graphics International (CGI'99)*, pages 214-221, 1999.

[VT00] P. Volino and N. Magnenat-Thalmann. *Virtual Clothing Theory and Practice*, Springer-Verlag, 2000.

[WMT05] H. Wang, P. J. Mucha, and G. Turk. Water Drops on Surfaces. In *Proceedings of ACM SIGGRAPH* 2005, Los Angeles, California, July 2005.

[WMW86] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, Vol. 2, No. 4, 227-234, February, 1986.

[YJC99] Young-Jung Yu, Ho-Youl Jung, and Hwan-Gue Cho. A new water droplet model using metaball in the gravitational field. *Computers & Graphics*, Vol. 23, No. 2, 213-222, 1999.