國立交通大學

多媒體工程研究所

碩 士 論 文

以 粒 子 為 基 礎 的 蠟 燭 融 化 模 擬

A Particle-based Simulation for Candle Melting

研 究 生：趙修範

指導教授：莊榮宏　教授

林文杰　教授

中 華 民 國　一 百 零 一　年 十 二 月

以粒子為基礎的蠟燭融化模擬
A Particle-based Simulation for Candle Melting

研 究 生：趙修範　　Student：Hsiu-Fan Chao

指導教授：莊榮宏　　Advisor：Jung-Hong Chuang

林文杰　　Wen-Chieh Lin

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

December 2012

Hsinchu, Taiwan, Republic of China

中華民國 一百零一 年 十二 月

以粒子為基礎的蠟燭融化模擬

研究生 ： 趙修範　　指導教授 ： 莊榮宏 博士

林文杰 博士

國立交通大學

資訊學院多媒體工程研究所

摘 要

我們提出一個基於光滑粒子流體動力學法(Smoothed-Particle Hydrodynamics)來模擬蠟燭融化。藉由使用幽靈粒子(ghost particle)並且確保橫跨固體邊界的密度場是連續的，融化後的液體可以自然地沿著表面流動，而不需要使用人工的界面張力。除此之外，藉由設定特別的幽靈速度(ghost velocity)以及使用平均後的速度場(XSPH)，流體的附著度可由一個直覺的參數調整。為了模擬融化的過程，一個完整的熱傳導被提出。熱傳導系統不止模擬粒子間的熱流，還模擬了和環境的熱交換。相變包含了固體到液體以及液體到氣體，它的發生是由模擬出來的溫度所決定。我們分裂粒子來保留黏滯液體因流動所產生的線狀物。螢幕空間顯像被用來呈現模擬結果。為了捕捉輪廓鮮明且凸出的液滴形狀，本來用於代表一顆粒子的球被拉長成橢球。

A Particle-based Simulation for Candle Melting

Student: Hsiu-Fan Chao

Advisor: Dr. Jung-Hong Chuang

Dr. Wen-Chieh Lin

Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

We propose a SPH-based particle simulation for candle melting. By using ghost particles and ensuring the continuous density field across the solid boundary, the melting liquids can flow along the surface naturally without applying artificial interfacial tension. Moreover, by setting specific ghost velocities and applying XSPH, the stickiness of the fluid can be controlled easily by an intuitive parameter. To simulate the melting process, a complete thermal transfer is proposed. The thermal transfer system simulates not only the heat flow between particles but also the heat exchange with the environment. Also, the occurrence of the phase transition, including solid-to-liquid and liquid-to-air, is determined by the simulated temperature. We split the particles to preserve the shape of thread formed by the flow of viscous liquid. Screen space point splatting method is applied to display the simulation result. The sphere originally used for representing the particle is stretched to the ellipsoid to capture the sharp and protruded shape of droplets.

Acknowledgments

I would like to thank my parents for their finance support and love.

Contents

List of Figures

CHAPTER 1

Introduction

Particle-based simulation has been an active research topic for many years and has received a great deal of attention. Different from grid-based simulation which is an alternative of physically-based simulation approach, particle-based simulation is suitable for simulating deformable objects, such as clothes, elastic objects , and melting materials. In this thesis, we focuses on the simulation of candle melting behavior, which is a common phenomenon in our daily life.

Although several methods for object melting simulation have been proposed, some problems require further study. First, the flow of melting liquid on the solid surface has been less studied. When the solid candle melts, the melting liquid needs to flow along the surface smoothly. Second, because the melted wax is a viscous material, it usually forms a thread on the way it flows. Third, most of the previous thermal transfer systems consider only the heat diffusion within material and do not simulate the heat exchange with the environment, which is important for candle melting. Finally, the shape of flowing viscous droplets on the surface is usually sharp and protruded, which might be smoothed by the current fluid surface reconstruction.

In this thesis, we develop a framework for candle melting and rendering. To deal with the solid boundary condition, we consider the solid particles as ghost particles that can contribute to the density summation. In addition, we sample air ghost particles near the free surface of the fluids. By making the density field continuous across the solid boundary, the melting liquids could flow on the surface naturally without the involvement of artificial interfacial tension. Moreover, by setting specific ghost velocities for solid particles and applying XSPH, the stickiness of the melting liquids could be controlled easily by using an intuitive parameter.

A complete thermal transfer system is proposed. It simulates not only the heat flow between particles but also the heat exchange with the environment. The phase transition from solid to liquid is simulated by varying the viscosity of the material with the simulated temperature. We also remove the particles that are overheated to simulate the phase transition from liquid to air.

To preserve the shape of thread formed by the flow of melting liquid, we propose a framework of particle splitting that is physically-inspired. The split criteria simply consider the forces exerted on the particle and do not require complex mathematical analysis. The result shows that we could preserve the thread properly. Besides, to intuitively model the shape of the thread, split particles are placed behind, instead of symmetrically around the original one.

We render the fluid using screen-space rendering with Translucent Shadow Map (TSM) for obtaining interactive frame rate. To capture the protruded and sharp detail of droplets, we stretch the sphere into the ellipsoid for representing particles.

1.1 Contribution

The contributions of the proposed candle melting framework can be summarized as follows:

- By utilizing the concepts of ghost particles and making the density field continuous across the solid boundary, the melting liquids could flow along the solid surface naturally without the need of applying artificial interfacial tension. Also, the stickiness of the fluids could be controlled by using an intuitive parameter.

- Propose a physics-inspired particle split operation for preserving the thread formed by the flow of melting liquid. Because the split criteria are physically-based that do not involve complex mathematical analysis, such as Principal Component Analysis (PCA), the computation is more efficient.

- Capture the sharp and protruded shape of the melting liquid in the process of rendering, instead of increasing the amount of particles involved in the simulation. By stretching the sphere into the ellipsoid and setting appropriate orientation, the detail of droplets can be better revealed.

1.2 Organization of the thesis

The rest of the chapters are organized as follows. Chapter 2 gives a literature review for SPH fluids, melting and flowing, and related particle-based fluid rendering approaches. Chapter 3 illustrates the proposed method, including the particle-based simulation framework, thermal transfer simulation in fluid elements, adaptive particle sampling scheme in object melting, and the screen space rendering approach. Chapter 4 shows the experimental results of our approach. Lastly, we summarize a conclusion and discuss the future work in chapter 5.

CHAPTER 2

Related Work

In this chapter, we briefly review the related works in several parts: SPH-based fluids, melting and flowing, and particle-based fluid rendering.

2.1 SPH-based Fluids

The Smoothed Particle Hydrodynamics(SPH) method, originally developed for simulating the astrophysical problem [Mon05], was first used by Desbrun et al. [DC96] in computer graphics field to animate highly deformable objects. This method can handle complex and large topological deformation and satisfies the mass conservation equation easily because it is inherently a lagrangian-based simulation approach.

Later, Müller et al. [MCG03] introduced a SPH-based fluid simulation scheme for interactive applications by deriving the equation of pressure, viscosity and surface tension term, and using the appropriate smoothing kernels (Figure 2.1). Since then, the development of largrangian-based fluid simulation, especially using SPH, has made great progress.

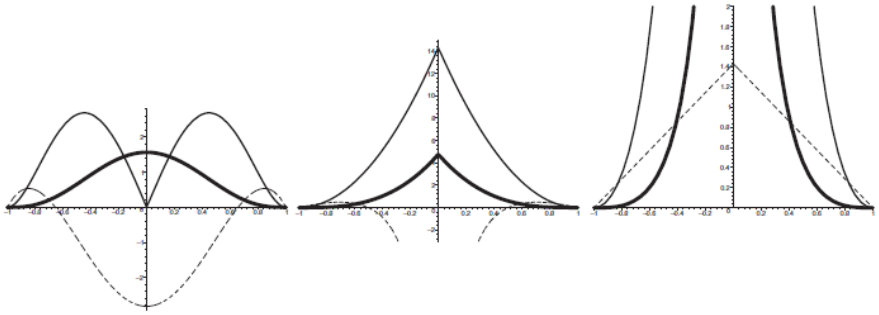


Figure 2.1: Three smoothing kernels W_{poly6}, W_{spiky} and $W_{viscosity}$ (from left to right). The thick lines are the kernels, the thin lines are the gradients in the direction towards the center, and the dash lines are the Laplacian [MCG03].

However, how to ensure the incompressibility of fluids in particle-based simulation is a challenge problem. In [MCG03], [BT07], and [APKG07], the particle pressure is determined by an equation of state(EOS). Solenthaler et al.[SP09] enforced incompressibility by purposing a prediction-correction scheme to solve particle pressure. Their method not only avoided the computational cost of solving a pressure Poisson equation but also enlarged the simulation time step. Hence, they achieved an order of magnitude speed up for the entire fluid animation. Raveendran et al. [RWT11] achieved the incompressibility of particle-based fluid animation by solving the Poisson equation on the coarse grid to enforce a divergence free velocity field.

Besides, the discretization resolution limitation is another problem because low resolution animation often limits the amount of detail that could be represented. The computation time and the simulation quality are always the trade-off. Thus, how to allocate more particles at the region of interest becomes a popular research topic. Adams et al. [APKG07] split and merged particles based on the extended local feature size(ELFS) defined on each particle. They split the particle if its ELFS is low, which indicates that this particle is near the fluid surface. Vice versa, they merged those particles whose ELFSs are high, which indicates that the particles are inside the fluid volume or near a thick flat

surface. Solenthaler et al. [SG11] proposed a two-scale method for particle-based fluid simulation. By simulating particles with the same size within each simulation level and applying the feedback force between them, Solenthaler et al. avoided the operation of particle split and merge and allowed the simulation of particles in large size differences without introducing the stability problem. Ando et al.[ATT12] preserved fluid sheets of animated liquids with an adaptively sampled method. They first detect the thin sheet area by analyzing the distribution of neighbor particles. This process generally follows by the algorithm for determining the stretch and the orientation of anisotropic kernels proposed by Yu et al. [YT10]. Ando et al. use it as a criterion to determine whether a particle is located at the thin sheet. After extracting the thin sheets, they detect the candidate positions where the particles should be placed. See Figure 2.2.

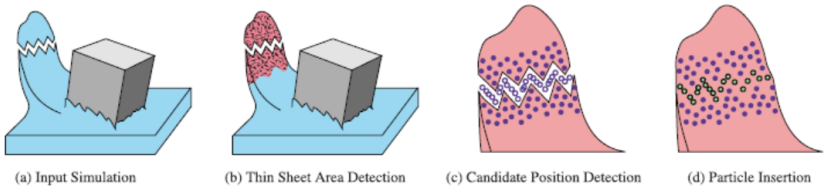


Figure 2.2: Preserve thin sheets by inserting the particles[ATT12].

2.2 Melting and Flowing

To simulate object melting using particle-based approach, we have to pay attention to the simulation of thermal dynamics and the interaction between the solid surface and the melting liquid. The first issue can be deal with intuitively, and no matter the simulation is grid-based or particle-based. The second has not received much attention in the current object melting simulation. Several methods [CMRBVHT02][PPLT06][PPLT09][SSP07] proposed for simulating the object melting do not address the flow of melting liquid on the surface of the solid object. In the following paragraphs, we briefly review the previ-

ous methods that are related to the object melting simulation.

Carlson et al. [CMRBVHT02] modified the Marker-and-Cell (MAC) algorithm to simulate the high viscosity materials, introduced a heat diffusion equation to handle the change in temperature, and varied the material's viscosity according to the temperature for simulating the phase transition process. Paiva et al. [PPLT06] [PPLT09] simulated the non-Newtonian fluids by using the jump number, which is a new rheological parameter of a viscoplastic fluid and determines the viscosity of the material. Similar to the method proposed in [CMRBVHT02], Paiva et al. only took the heat diffusion into consideration and interpolated the jump number according to the temperature to simulate the process of the phase transition. In [SSP07], Solenthaler et al. proposed a unified particle model for fluid-solid interaction. They only considered the heat diffusion between particles and linearly interpolate Young's Modulus and the viscosity to simulate the phase transition. However, they do not handle the particle penetration problem.

Iwasaki et al. [IUDN10] simulated the ice melting and focused on the flows and the motion of the melt liquids on the solid as well. They took not only heat diffusion but also the heat radiation into consideration. The radiation energy from the heat source is assumed to be transfered by a set of photons [FM07]. However, the radiation energy carried by the photons do not attenuate with the distance and the thermal dissipation is not considered. Besides, Iwasaki et al. used a simple and artificial force model to simulate the interfacial tension between water-water and water-ice particles; see Figure 2.3. With these interfacial attraction forces, the meltwater could form a thin film of water that surrounds the ice and form a water droplet at the bottom of the ice. However, it needs additional parameters tunning to obtain the desired result.

Maréchal et al. [MGG+10] proposed several thermal transfer rules based on physics to synthesize realistic winter sceneries. Three types of heat transfer are considered: conduction, convection, and radiation. Figure 2.4 demonstrates the heat transfer model.

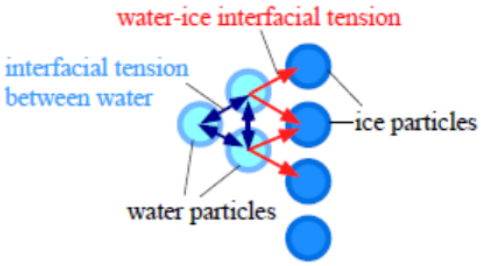


Figure 2.3: Interfacial tension in [IUDN10].

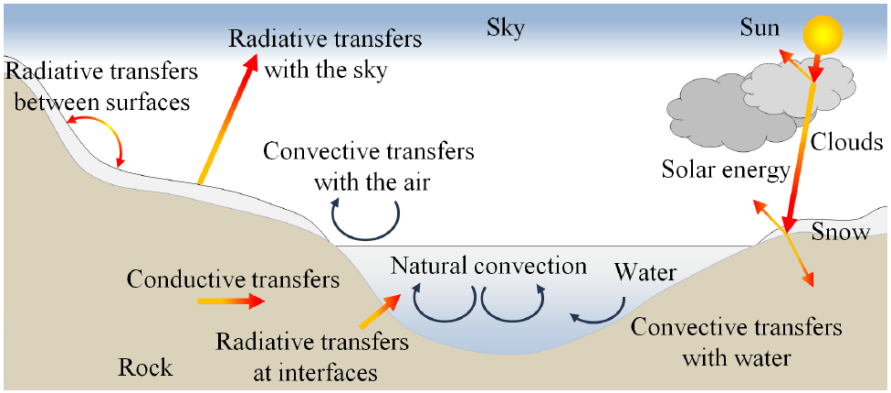


Figure 2.4: Thermal transfer in [MGG+10].

2.3 Rendering

Nowadays, marching cube may be the most well-known surface reconstruction method for fluid rendering. To generate the triangular mesh for particle-based fluids, we need to first construct the density field of the particles, and then find the iso-surface using the marching cube algorithm. The major concern of this approach is that the quality of marching-cube reconstruction depends on the grid resolution and the preciseness of density field constructed.

Researchers try to either speed up or improve the surface reconstruction algorithm. Akinci et al. [AIAT12] speeded up the algorithm by applying the marching cube algorithm only near the band of the fluid surface. Ho et al. [HWC+05] solved the ambiguity

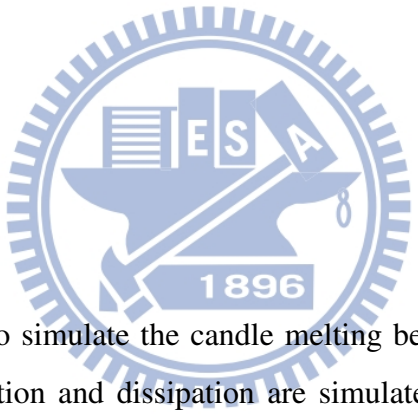
problem by converting 3D marching cubes into 2D cubical marching squares. Yu et al. [YT10] analyzed the distribution of the nearby particles by using weighted principle component analysis (WPCA), and applied the anisotropic kernel for each particle to construct the density field to better reveal the particle density distribution. Although improvement have been made, the marching cube algorithm cannot be performed in realtime.

Screen space point splatting method proposed by Cords et al. [CS08] may be the fastest particle-based fluid rendering approaches because they only generated the surface visible to the camera and eliminated the requirement of mesh generation [MSD07]. Cords et al. generated a 2D smooth depth map from the visible particles and the surface normals of the fluid are obtained from the smoothed depth map. Finally, the surface shading is carried out in screen space by using the surface normal and the depth obtained from the previous steps. Moreover, screen space point splatting method is easy to be post-processed so that we could remove some artifacts or enhance the detail that is not captured in simulation. Cords et al. [CS08] used a Binomial filter to smooth the fluid surface. van der Lann et al. [vdLGS09] smoothed the surface by minimizing the curvature and additionally applied the noise texture to enhance its quality.

However, differed from rendering a bulk of fluid, such as water, the subtle and sharp shape of the melting liquids and the droplets need to be captured carefully. In [CS08] and [vdLGS09], they approximated the smooth fluid surface by rendering particles as spheres and did not emphasize the sharp shape of droplets. In [vdLGS09], they used Perlin noise to add the foam-like effects that do not appear in object melting.

CHAPTER 3

Candle Melting Framework



3.1 Overview

The aim of this study is to simulate the candle melting behavior naturally. First, the thermal radiation, conduction and dissipation are simulated and the phase transition from solid to liquid will take place when the temperature reaches the melting point, and the liquid particles near the candle flame transits to the air when they are burned. Second, the melting liquid usually moves along the solid surface smoothly and forms a thread on the solid surface, and in some cases leaves the solid object and falls down due to gravity.

Figure 3.1 is the flow of our candle melting algorithm. First, giving a candle model, we sample it with a set of particles and compute neighbors for each particle. Then, we compute the temperature of each particle by simulating the heat transfer between particles. Three types of thermal transfer are considered: thermal radiation, thermal

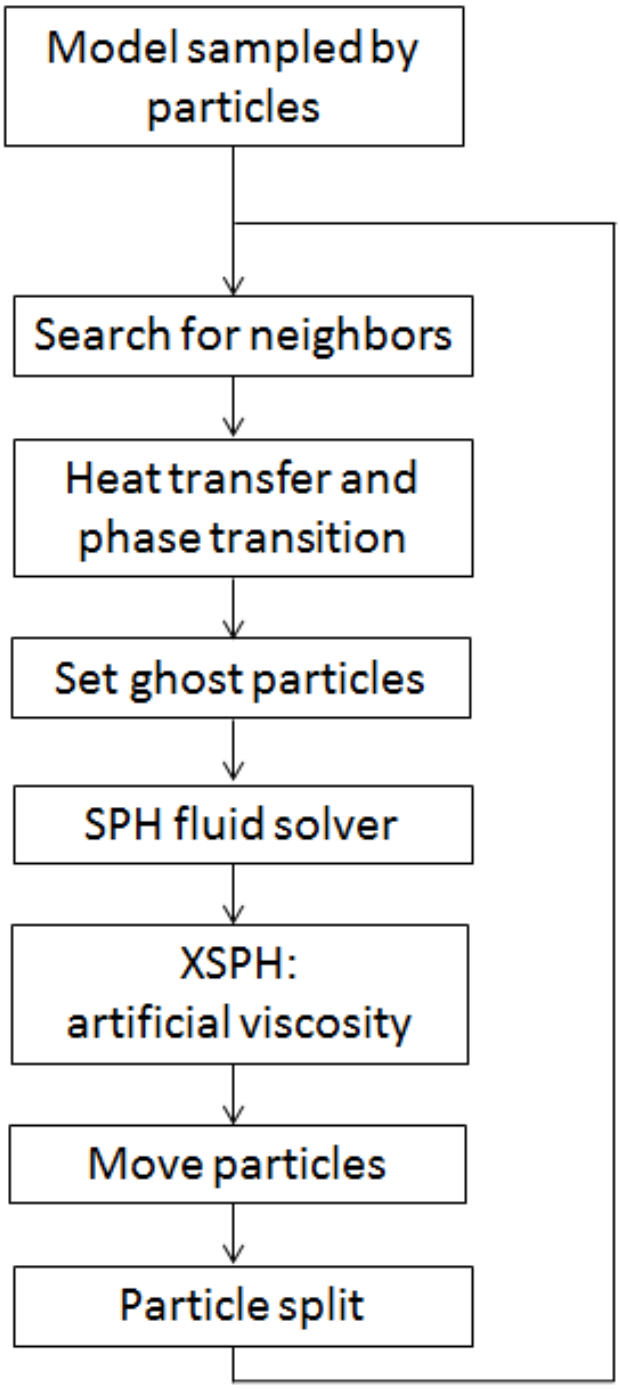


Figure 3.1: System flow of our algorithm.

conduction, and thermal dissipation. Once the temperature of a particle reaches the melting point, the phase transition from solid to liquid takes place and the viscosity of melting liquid also varies according to the temperature. Moreover, to simulate the combustion, the liquid transits to the air when it reaches the flash point.

To reveal the realistic cohesion between melting liquids and solids and prevent liquid particles from penetrating the solid, we consider solid particles as ghost particles, and assign some elaborate attributes to them. We also compensate density deficiency near the free surface by sampling air particles that are viewed as ghost particles. The fluid particles thus can distribute evenly on the solid surface instead of being clustered to a shell. After solving Navier-Stokes equation using SPH, we cloud determine the velocity of each particle. However, in order to damp off the unphysical particle oscillation and allow the melting liquids flow smoothly on the solid surface, we further smooth the velocity of each liquid particle by applying XSPH to nearby particles. Besides, to simulate the thread formed by melting liquids, we also split the liquid particle dynamically.

We use screen space rendering approach to display the simulation result. To model the shape of viscous melting liquids flowing on the surface, we stretch the particle's sphere to the ellipsoid. The stretchiness and the orientation of the ellipsoid are different according to the shape we want to model.

3.2 Fluid Simulation Framework

In this section, we first describe the framework of the fluid solver, and how we formulate the force exerted on the fluids by using SPH. Then, we explain the concept of using ghost particle in the object melting simulation.

3.2.1 SPH formulations and Navier-Stokes equation

SPH is an interpolation method defined on the particle system. With SPH, fluid attributes could be evaluated at discrete particle positions by using a symmetric radial function. According to SPH, a scalar quantity A at location r could be interpolated by a weighted sum of contributions from the neighbor particles:

$$A(r) = \sum_j A(r_j) \frac{m_j}{\rho_j} W(\mathbf{r} - \mathbf{r_j}, h), \tag{3.1}$$

where j iterates over neighbor particles, m_j is the mass of the particle j, r_j is the location of the particle j, ρ_j is the density define on the particle j, $A(r_j)$ is the quantity at r_j. The function W is a symmetric radial function with the kernel size h. We compute the density at r by using

$$\rho(\mathbf{r}) = \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r} - \mathbf{r_j}, h) = \sum_j m_j W(\mathbf{r} - \mathbf{r_j}, h), \tag{3.2}$$

which is also a derivation of Equation 3.1.

With SPH, the derivatives of field quantities only affect the smoothing kernel, so the gradient and Laplacian of A at location r could be derived easily by Equation 3.3 and 3.4

$$\nabla A(r) = \sum_j A_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r_j}, h) \tag{3.3}$$

$$\nabla^2 A(r) = \sum_j A_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r_j}, h) \tag{3.4}$$

Fluids are often described by a velocity field $\mathbf{v}$, density field ρ, and pressure field p. The evolution of these quantities over time is governed by two equations. The first is mass conservation equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{3.5}$$

The second is Navier-Stokes equation which formulates the conservation of momentum

$$\rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}, \tag{3.6}$$

where μ is the viscosity of the fluid, and $\mathbf{g}$ is external force field. Because we use the particle system to model the entire fluid, both of these two equations could be further simplified. First, because the number of particles is constant and each particle has a constant mass during the whole simulation process, mass conservation is guaranteed naturally. Second, since the particles move with the fluid, the convective term $\mathbf{v} \cdot \nabla \mathbf{v}$ is not needed for the particle system. That is,

$$\rho(\frac{\partial \mathbf{v}}{\partial t}) = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}. \tag{3.7}$$

There are three forces on the right hand side of the Equation 3.7 modeling pressure, viscosity, and external forces. We could formulate these force density terms acting on a particle i using SPH as follows [MCG03]:

$$\mathbf{f}_i^{pressure} = -\nabla p = -\sum_j m_j \frac{(p_i + p_j)}{2\rho_j} \nabla W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{3.8}$$

$$\mathbf{f}_i^{viscosity} = \mu \nabla^2 \mathbf{v} = \mu \sum_j m_j \frac{(\mathbf{v}_j - \mathbf{v}_i)}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{3.9}$$

To compute the pressure term in Equation 3.8 for each particle, we use ideal gas state equation

$$p_i = k(\rho_i - \rho_o), \tag{3.10}$$

where k determines the incompressibility of the fluid, ρ_i is the density defined on the particle i, and ρ_o is the rest density of the fluid [DC96].

After computing all the forces exerted to the particle i, the acceleration of it can be computed by using Equation 3.11.

$$\mathbf{a}_i = \frac{\mathbf{f}_i}{\rho_i} = \frac{\mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscosity} + \rho_i \mathbf{g}}{\rho_i}, \tag{3.11}$$

where $\mathbf{a}_i$ is the acceleration of particle i. The velocity and the position of the particle i is then updated using Equation 3.12 and 3.13.

$$\mathbf{v}_i^* = \mathbf{v}_i + \mathbf{a}_i \Delta t, \tag{3.12}$$

where $\mathbf{v}_i^*$ is the updated velocity and Δt is the time interval, and

$$\mathbf{r}_i^* = \mathbf{r}_i + \mathbf{v}_i^* \Delta t, \tag{3.13}$$

where $\mathbf{r}_i^*$ is the updated position.

3.2.2 Smooth flow of melting liquids

To reveal the cohesion phenomenon between solid and melting liquid and let the melting liquid flows smoothly on the solid surface, we apply the concept proposed by Schechter et al.[SB12] in our object melting simulation. First, by using ghost particles near fluid boundary, we prevent the density of a liquid particle from being under-estimated, which introduces obvious artifacts in SPH simulation. Second, by making the density field continuous across the solid boundary, we could reach no-penetration and no-separation solid-liquid boundary condition easily. Finally, we apply an artificial viscosity to damp off the non-physical oscillations and obtain smooth fluid behavior when the melting liquid flow on the solid surface.

Ghost particle Obviously, the fluid density distribution computed by Equation 3.2 is not even because the amount of neighbors between interior and boundary liquid particles differs. The pressure equation (Equation 3.10) leads to a case that particles near the boundary are clustered to a shell, aiming to rebalance the density; see the Figure 3.2 .

To correct the density summation for a fluid particle near the boundary of the fluid, we first consider solid particles as ghost particles with mass equal to the liquid particle and also contribute to the density summation for liquid particles. As a result, we prevent liquid particles near the solid boundary from being clustered to a shell due to the

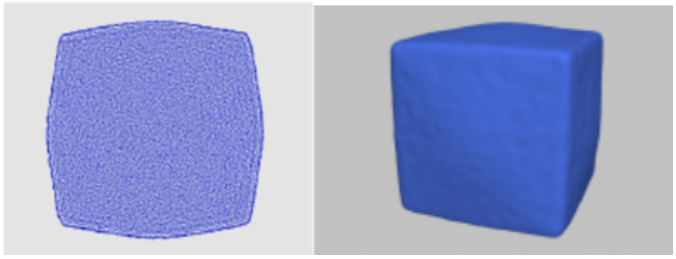


Figure 3.2: The fluid particles clustered into a shell due to neighbor deficiency [SB12].

neighbor deficiency. Also, we add air ghost particles above the free surface of the fluid and within the kernel size [SB12]. However, for simplicity, the position of an air ghost particle for the liquid particle is set to be the reflection of some solid particles; as shown in Figure 3.3.

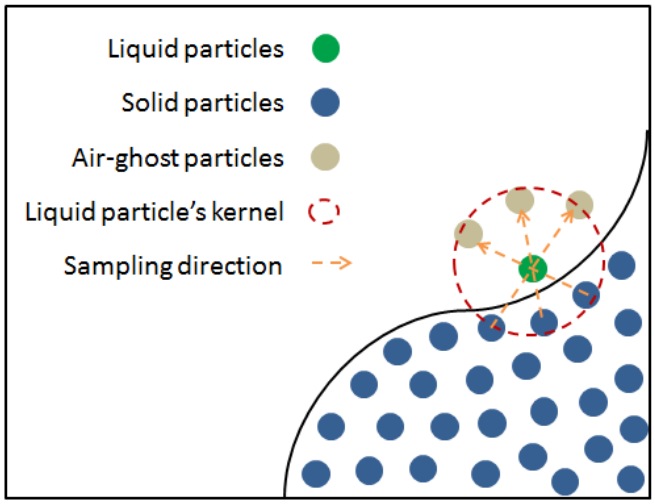


Figure 3.3: Sample air ghost particles.

Besides, to prevent liquid particles from either penetrating or leaving solid surface, the pressure field should be continuous through the boundary. Thus, after computing the density for each liquid particle, we set the density of each solid particle equal to the density of it's nearest liquid particle. This step naturally enforces the no-penetration and no-separation boundary condition between solid and liquid.

XSPH To damp off the unphysical oscillation when the fluids flow on the solid surface, we apply the artificial viscosity to diffuse the velocity with nearby particles by Equation 3.14 :

$$\mathbf{v}_i^{new} = \mathbf{v}_i^* + \epsilon \sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j^* - \mathbf{v}_i^*)W(\mathbf{r} - \mathbf{r_j}, h), \tag{3.14}$$

where j iterates over neighbor particle including solid particles, $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$ are the velocities of particle i and j after solving the Navier-Stokes equation, ϵ is a parameter that represents the degree of diffusion [SB12].

However, before we apply the artificial viscosity, we set elaborate ghost velocity for solid particles. The velocity of liquids near solid boundary thus tends to converge to that ghost velocity after the computation of Equation 3.14 and achieves specific stickiness. For the inviscid liquid, it often freely slips on the solid boundary because the normal component of liquid and solid velocities match and the motion of the liquid is only determined by its tangential velocity. Thus, Schechter et al. [SB12] constructed the ghost velocity $\mathbf{v}^{ghost}$ for a solid particle by summing the normal component of particle's true velocity $\mathbf{v}_N^{solid}$ and the tangential component of the nearest liquid particle's velocity $\mathbf{v}_T^{liquid}$, as shown in Equation 3.15.

$$\mathbf{v}^{ghost} = \mathbf{v}_N^{solid} + \mathbf{v}_T^{liquid}. \tag{3.15}$$

Obviously, by constructing the ghost velocity for solid particle as Equation 3.15, the liquid near the solid boundary could achieve no-stick boundary condition because there does not exist relative-velocity in the normal direction and the velocity of the liquid does not suffer any friction in the tangential direction.

However, we modify the Equation 3.15 as the following:

$$\mathbf{v}^{ghost} = \mathbf{v}_N^{solid} + \alpha\mathbf{v}_T^{liquid}, \tag{3.16}$$

where α, $0 \leq \alpha \leq 1$, is a tunable tangential friction coefficient that allows us to achieve the desired fluid stickiness more easily. For the very high viscous liquid in our case, we let α approach to zero such that the liquid could almost stick on the solid surface.

3.2.3 Velocity Update

Figure 3.4 shows the detail steps for computing the velocity of the melting liquids. We first compute the density for each liquid particle by using Equation 3.2. In this step, both the solid ghost particles and the additionally sampled air ghost particles contribute to the density summation. Then, we set the density of each solid particle as the density of its nearest liquid particle, expecting that the pressure computed in the next step could be continuous across the boundary between solid and liquid. After computing the pressure using Equation 3.10 for each particle, we could compute the forces exerted on the liquid particle by using Equation 3.8 and Equation 3.9. Then, we solve the velocity for melting liquids by Equation 3.12. Finally, we set the ghost velocity for solid particle using Equation 3.16 and smooth the velocity of liquid particles by applying XSPH (Equation 3.14).

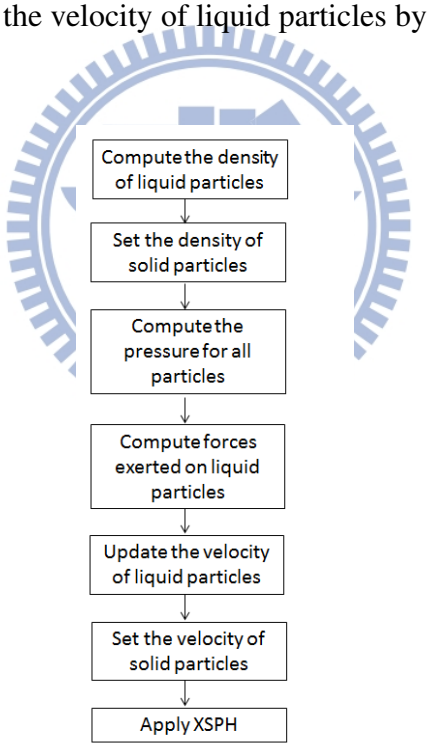


Figure 3.4: Flow for updating the velocity of melting liquids.

3.3 Thermal Transfer System

Thermal transfer system plays an important role in simulating the melting object because the melting behavior is triggered according to the simulated temperature. Figure 3.5 shows each step of our heat transfer system. To determine the temperature of each

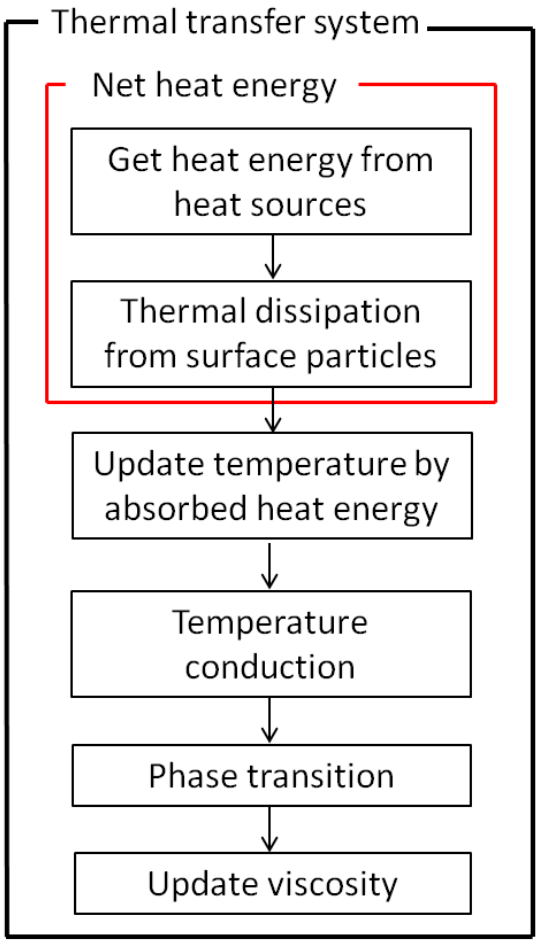


Figure 3.5: Thermal transfer system flowchart.

particle, we first need to simulate the heat flow of our system. We take three types of thermal transfer, namely, thermal radiation, thermal dissipation, and temperature conduction, into account. After simulating the net heat energy flow in the system, the

absorbed heat energy Q_i of the particle i is

$$Q_i = Q_i^{radiation} - Q_i^{emission}, \tag{3.17}$$

where $Q_i^{radiation}$ and $Q_i^{emission}$ is the radiation energy received and emitted by particle i respectively. The temperature T_i of particle i is then updated by ΔT_i defined as

$$\Delta T_i = \frac{Q_i}{m_i C}, \tag{3.18}$$

where m_i is the mass of the particle, and C is the specific heat capacity of the material. The updated temperature is thus conducted to other particles in the entire model and the occurrence of the phase transition could be determined as well. The phase transitions considered are not only from solid to liquid but also liquid to air. The solid transits to the liquid if the temperature is higher than the melting point. For the overheating liquid particles, they transit to the air and do not involve in the simulation anymore.

In reality, the viscosity of the material often varies from the temperature to temperature. Hence, we linearly interpolate the viscosity coefficient μ in Equation 3.9 for each liquid particle according to its temperature. See the Figure 3.6. The simulated viscosity force in Navier-Stokes equation (Equation 3.6) thus influences the motion of the melting liquids.

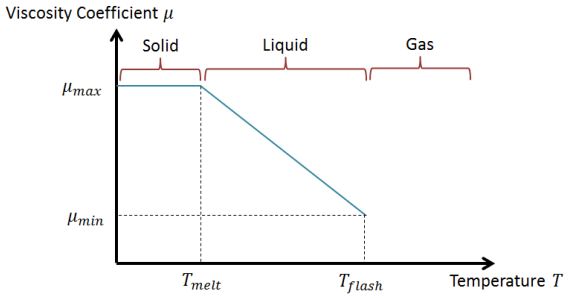


Figure 3.6: Interpolating viscosity coefficient.

Thermal Radiation For the particles heated by the heat source directly, they would absorb the radiation energy. To efficiently locate the particles that can be heated by the

heat sources, we apply the shadow map technique to find the particles that are visible to the heat sources. Moreover, the radiation energy from the heat source often attenuate with the distance due to the occlusion of the air. For simplicity, we formulate the radiation energy received by particle i that is at location $\mathbf{r}$ and visible to some heat sources as the following:

$$Q_i^{radiation} = \sum_s \delta Q_s \Delta t W(\mathbf{r} - \mathbf{p_s}, r_s), \tag{3.19}$$

where s iterates over the visible heat sources, δ is the coefficient that determines the heat absorption ratio, Q_s is the heat energy from the heat sources per unit time, Δt is the time interval, W is a weighting function, $\mathbf{p_s}$ and r_s are the position and the effective radius of the heat sources, respectively. In our cases, we use the spiky kernel as the weighting function.

Thermal Dissipation With the thermal dissipation, the liquid flowing on the solid surface has the chance to transit to solid again, forming rich shapes. For the surface particles exposed to the air, they either absorb or release the heat energy according to the difference between their temperature and the ambient temperature. We use the concept of color field proposed by Muller et al. [MCG03] to locate the surface particles and formulate the dissipation behavior using the following general thermal radiation equation proposed by [MGG+10].

$$Q = \epsilon_e \sigma A(T^4 - T_{ambient}^4), \tag{3.20}$$

where ϵ_e is the emissivity of the material, σ is the Stefan-Boltzmann constant, A is the area of heat transfer surface, T is the temperature of the voxel in their case, and $T_{ambient}$ is the ambient temperature. However, because it is not easy to compute the heat transfer surface for a particle exactly, we use the surface area of the particle as an approximation.

$$Q_i^{emissive} = \epsilon_e \sigma 4\pi h_i^2 (T_i^4 - T_{ambient}^4), \tag{3.21}$$

where h_i is the kernel size of the particle i, and T_i is the temperature of the particle i.

Temperature Conduction Once the temperature of each particle has been updated according to Equation 3.18, the temperature conduction between particles could be calculated using the following equation:

$$\frac{\partial T_i}{\partial t} = c_d \sum_j m_j \frac{T_j}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h), \tag{3.22}$$

where diffusion rate c_d influences the speed of thermal conductivity, T_j is the temperature of particle j, and W is the kernel function which is the same as the one used in computing viscosity force. Equation 3.22 is a SPH-form of Equation 3.23 that gives the change in temperature T.

$$\frac{\partial T}{\partial t} = k\nabla^2 T - (\mathbf{v} \cdot \nabla)T, \tag{3.23}$$

where k is called the thermal diffusion constant, and $\mathbf{v}$ is the fluid velocity [CMRBVHT02]. The derivation from Equation 3.23 to Equation 3.22 is simple. First, as we mentioned before (in Section 3.2.1), the convective term $(\mathbf{v} \cdot \nabla)T$ could be ignored in particle-based simulation. Then, Equation 3.22 could be derived by computing the Laplacian of the temperature $\nabla^2 T$ using Equation 3.4.

Similar to the concept of avoiding asymmetric viscosity forces distribution [MCG03], taking the temperature differences between particle i and its neighbor particle j into consideration is required.

$$\frac{\partial T_i}{\partial t} = c_d \sum_j m_j \frac{(T_j - T_i)}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h). \tag{3.24}$$

Finally, the temperature after conduction is determined by Equation 3.25.

$$T_i^* = T_i + \Delta t \frac{\partial T_i}{\partial t}, \tag{3.25}$$

where T_i^* is the updated temperature of particle i and Δt is the time interval.

3.4 Adaptive Sampling

So far we could simulate the motion of melting liquids naturally, but we haven't tried to capture their subtle shape and behavior. For the viscous liquid, it usually stretch to a thread when it flows. Similar to the concept of Ando et al.[ATT12], we adaptively sample the particle to preserve the thread formed by the flow of droplets.

Framework for Adaptive Particle Sampling During the simulation process, if the split criteria are satisfied for particle i (we explain the criteria in next paragraph), we split the particle i and sample new particles within the kernel radius of the particle i with lighter mass. Particle i is kept in the same position and those newly sampled particles are placed right behind it. These particles thus form a shape of thread on the surface.

We associate the particle i with a level l_i to record how many times the particle i has been split. l_i is set to zero originally and is updated to $l_i + 1$ after splitting. We compute the mass and kernel size of each particle using Equation 3.26 and 3.27 after each split process.

$$m_i = \frac{m}{(s+1)^{l_i}}, \tag{3.26}$$

$$h_i = \frac{h}{\sqrt[3]{(s+1)^{l_i}}}, \tag{3.27}$$

where s is the number of newly sampled particles, m and h are the initial particle mass and initial kernel radius respectively. Note that when a particle i at level l_i splits, the level of newly sampled particles is also set to $l_i + 1$ so that the mass conservation could be guaranteed. Figure 3.7 demonstrates how we update the level of particles after split prcoess. Figure 3.8 shows the mass of particles computed by using Equation 3.26. Theoretically, particle i that has been split can split as usual if the split criteria are guaranteed for it. However, in order to avoid over-splitting that might introduce the

unstable simulation, we do not split the particle i whose level l_i is greater than three even if the split criteria are satisfied.

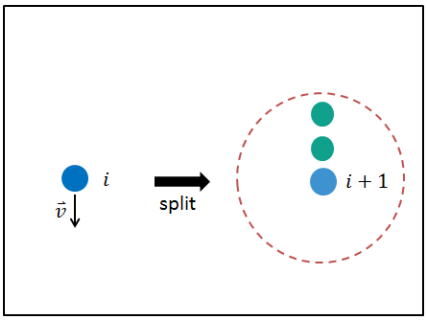


Figure 3.7: Case of s = 2. Splitting a particle of level i into three particles of level $i+1$. The newly sampled particles are placed behind the original one and within the kernel radius.

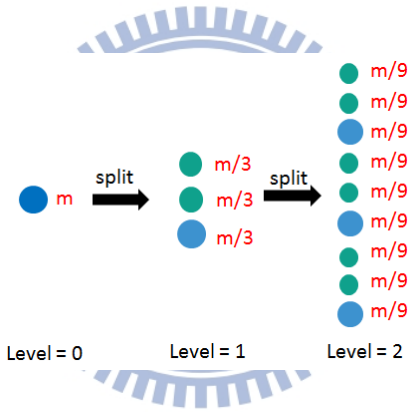


Figure 3.8: Case of s = 2. The change of mass for particles during the split process.

Other physical attributes of the newly sampled particles (e.g. conductivity, emissivity, temperature, viscosity coefficient, and velocity etc) are set to be the same as the original particle. For the particles with different kernel sizes and masses, the force model used in fluid simulation need to be further modified to avoid asymmetric force distribution. Hence, we use the force model as Adams et al. [APKG07].

$$\mathbf{f}_i^{pressure} = -\frac{m_i}{\rho_i}\sum_j \frac{m_j}{\rho_j}(p_i + p_j)\frac{(\nabla W(\mathbf{r_i} - \mathbf{r_j}, h_i) + \nabla W(\mathbf{r_i} - \mathbf{r_j}, h_j))}{2} \tag{3.28}$$

$$\mathbf{f}_i^{viscosity} = \mu \frac{m_i}{\rho_i} \sum_j \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) \frac{(\nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h_i) + \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h_j))}{2} \quad (3.29)$$

Particle Split Criteria Our concept is to detect which droplet is stretched to the thread and in that case we split that droplet and sample the new liquid particles right behind that droplet to model this shape. As shown in Figure 3.9, we observe that the thread is going to be formed when the droplet is stretched, and at that moment, the external force exerted on the mass of liquid is larger than the internal force from itself.

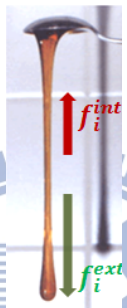


Figure 3.9: Real honey thread and the forces exerted on it.

Based on this observation, we split the particle i and sample particles behind it when the following two conditions are met:

- $\|\mathbf{f_i^{ext}}\| > \|\mathbf{f_i^{int}}\|$,
- $\mathbf{f_i^{ext}} \cdot \mathbf{f_i^{int}} < \mathbf{0}$,

where f_i^{ext} and f_i^{int} are the external force and internal force exerted on the particle i. In our simulation model, the external force f_i^{ext} includes the gravity force and the cohesion force which is modeled as the pressure force from ghost-solid particles. The internal force f_i^{int} includes the pressure force and the viscosity force caused by the liquid particles.

3.5 Rendering Framework

To achieve real-time rendering of the simulated result, we apply the screen space approach similar to [Gre10]. Moreover, to model the protruded and sharp shape of the thread formed by the liquid drops, we stretch the sphere that is usually used in the screen space rendering to the ellipsoid, vary the radius of the sphere based on its mass, and rotate it to fit the surface of the model.

3.5.1 Screen Space Rendering

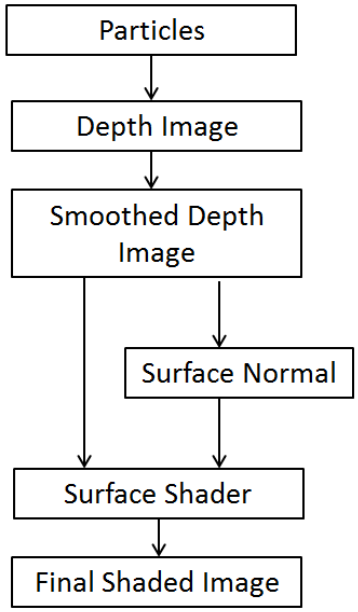


Figure 3.10: Flowchart of screen Space Rendering.

As shown in Figure 3.10, by rendering particles as spherical point sprites, we obtain a depth map of the particles viewing from the eye point. However, the depth map is often too bumpy to be used. Hence, we smooth the depth map by using Gaussian filter. Figure 3.11 shows the depth map with and without applying Gaussian filter.

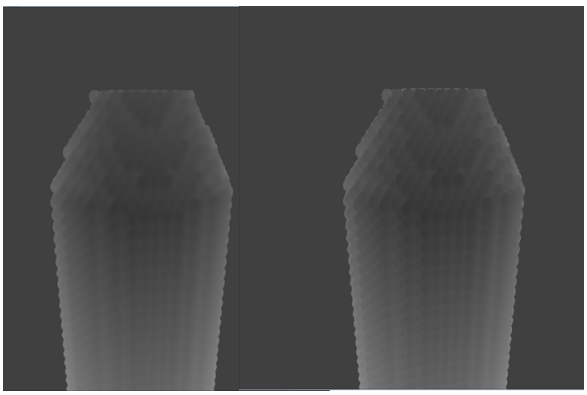


Figure 3.11: Depth map with/without Gaussian Filter.

After smoothing the depth map, we use it to compute the fluid normal vectors in per-pixel manner; see Figure 3.12. In addition, we also calculate the eye-space surface position from the smoothed depth map. Once we have the per-pixel information of the surface, such as depth, normal, and position, we could shade the surface as usual, such as by Phong shading. Finally, because there exist solidified particles on the object surface that often introduce self-shadows, we also apply shadow map to generate the shadow.

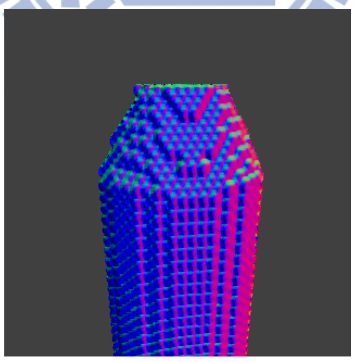


Figure 3.12: Normal map obtained from the depth map filtered with Gaussian filter.

3.5.2 Model the shape using ellipsoids

To model the shape of a thread formed by the melting liquid on the solid surface , we first render the spheres in different radius based on the particle's split level obtained in adaptive sampling. For the particle with less mass, we render it with a relatively small sphere, and vice versa, for the particle at lower split level that has heavier mass, we render it with a relatively large sphere. Figure 3.13 demonstrates the idea. Moreover, according to the suggestion of Cords et al. [CS08], the radius for rendering a particle is determined when approximating the surface of calm liquid and mainly depends on the distance to its nearest neighbor.

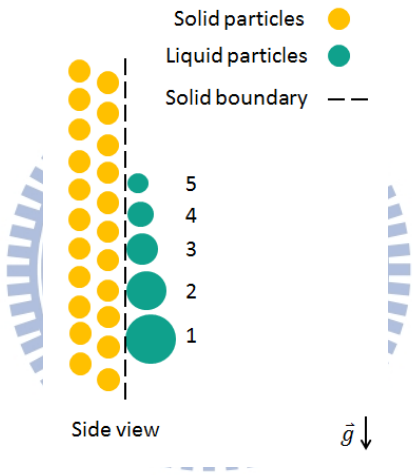
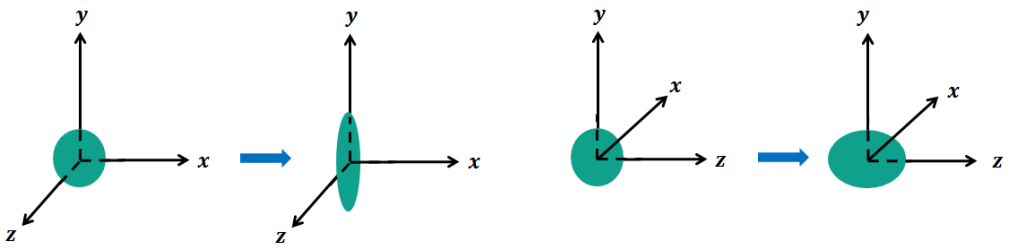


Figure 3.13: Render particles using spheres in different radius. The number indicates the particle's split level.

Then, we stretch the sphere in specific direction, trying to model the protruded shape of the thread. For the sphere at the bottom of the thread, we stretch it in the direction such that a sharp prominence could be formed at the bottom of the thread. For the sphere located at the middle of the thread, we stretch the sphere in the direction to make the filament appearance. The stretchiness of the sphere is an empirical adaption that depends on the specific requirement. See the Figure 3.14.



(a) (b)

Figure 3.14: (a) Stretch the sphere in y-direction so that it looks like a filament. (a) Stretch the sphere in z-direction to make a sharp prominence.

Finally, we rotate the ellipsoid, and align its orientation with the surface normal vector obtained by computing the divergence of the color field [MCG03].

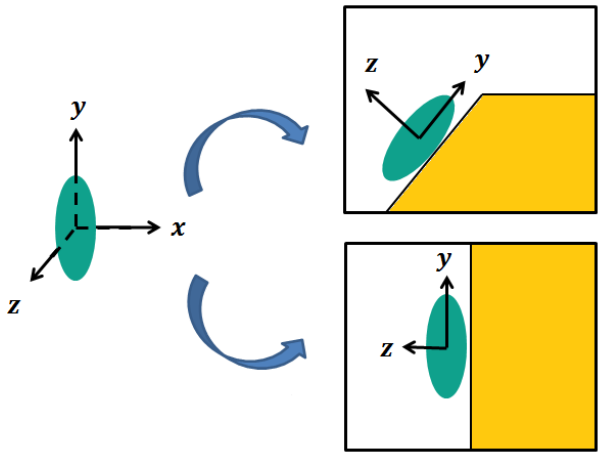


Figure 3.15: Rotate the ellipsoid to align the surface normal vector.

3.5.3 Translucent Rendering

Because the wax is a translucent material, we integrate Translucent Shadow Map [DS03] in our rendering framework. Figure 3.16 shows the standard process of Translucent Shadow Map.

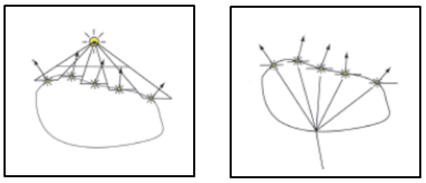


Figure 3.16: The Translucent Shadow Map stores irradiance samples (left). The radiance leaving the object is them computed by filtering these radiance samples (right) [DS03].

In first step, we view the candle flame as the light source and place a virtual camera at this position. The virtual camera looks at the particles and the shadow map stores irradiance samples. In the second step, we compute the radiance leaving the object and use it to shade the object. However, to get better visual effect and prevent the detail of droplets from looking blurred due to the strong translucent effect, we additionally consider the local light source to make the shaded droplet stereoscopic.

CHAPTER 4

Results

In this chapter, we first show the value of simulation parameters used in our results and how we render the results. Second, we demonstrate the effect of parameter that determines the stickiness of the fluid. Third, we show the thread-preserving flow of melting liquid by applying the adaptive particle sampling model. Then, we demonstrate the rendering result using screen space rendering approach. Finally, we compare our candle melting result with previous methods [CMRBVHT02][PPLT09].

Table 4.1 shows the value of simulation parameters used in the following results except Figure 4.1, 4.2, and 4.3. Table 4.2 show whether the technique in each column is applied in the following figures.

parameters	meaning	value
k	liquid incompressibility (Eq. 3.10)	645000
μ_{max}	max viscosity coefficient (Eq. 3.9 and Fig. 3.6)	17000
μ_{min}	min viscosity coefficient (Eq. 3.9 and Fig. 3.6)	16000
ϵ	degree of velocity diffusion (Eq. 3.14)	0.72
α	tangential friction of solid surface (Eq. 3.16)	0.38
Q_s	heat energy from heat source (Eq. 3.19)	65
δ	heat absorption ratio (Eq. 3.19)	0.063
r_s	heat source effective radius (Eq. 3.19)	0.18
c_d	thermal conductivity (Eq. 3.24)	0.004
ϵ_e	thermal emissivity (Eq. 3.21)	0.001
s	split number (Eq. 3.26)	2

Table 4.1: Main parameters of candle melting simulation

	Translucent Shadow Map	Adaptive Particle Sampling	Render the particle as ellipsoid
Fig 4.1	o	o	o
Fig 4.2	o	o	o
Fig 4.3	o	o	o
Fig 4.4	o	Δ	Δ
Fig 4.5	x	o	Δ
Fig 4.8	o	o	o

Table 4.2: How we render the simulation results. The circle (o) means the technique in that column is applied and (x) vice versa. (Δ) means we show the comparison with and without applying the technique.

To illustrate that the parameter α appeared in Equation 3.16 determines the stick-

iness of the fluid, Figure 4.1, Figure 4.2, and Figure 4.3 show the simulation process with different α values. The value of other parameters used in these results are equal to those in Table 4.1. Obviously, by setting different α values, the stickiness of the fluid on the solid surface is different. Hence, we could achieve the expected stickiness easily by only tunning this parameter.

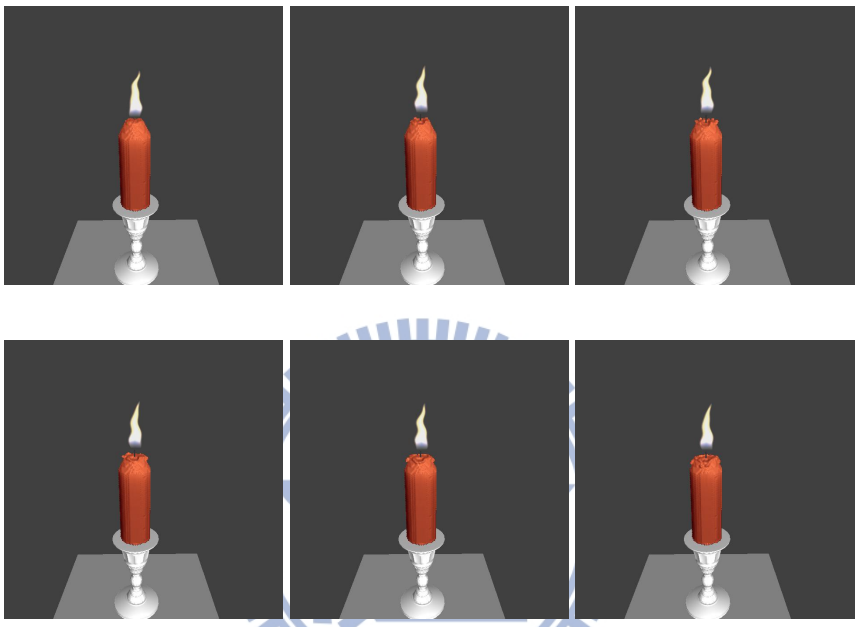


Figure 4.1: $\alpha = 0.2$. The fluid move slowly and almost stick on the solid surface.

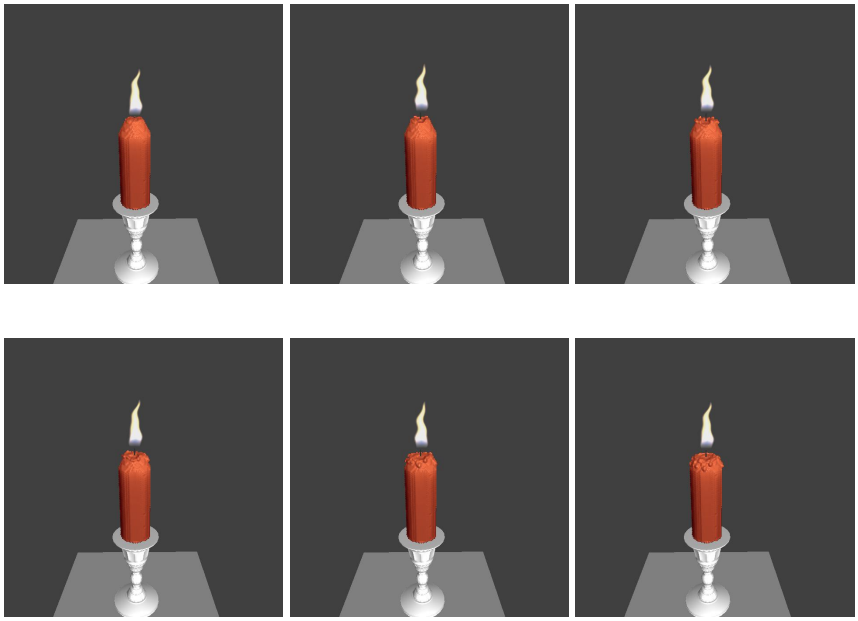


Figure 4.2: $\alpha = 0.5$. The fluid run along the solid surface smoothly.

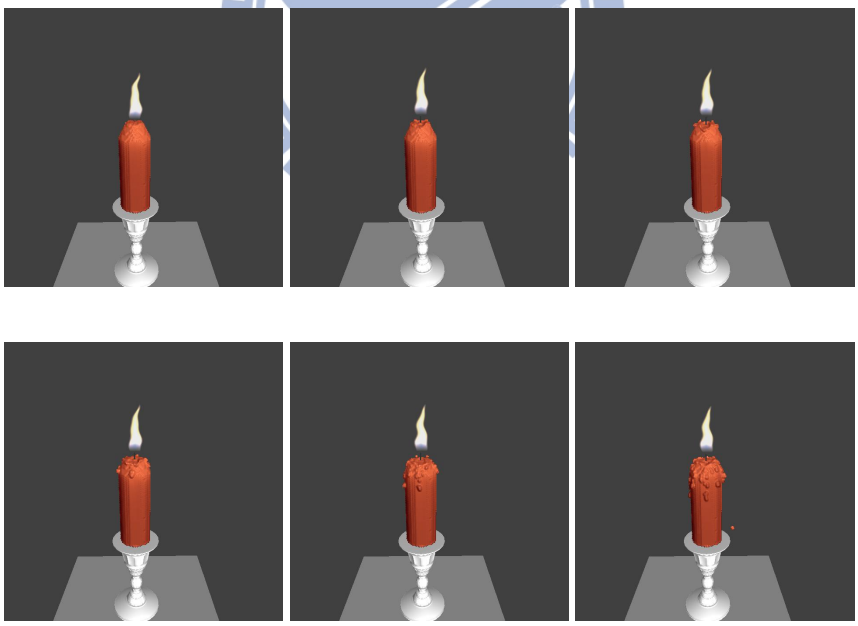


Figure 4.3: $\alpha = 0.8$. The fluid on the slope run away from the solid surface easily.

Figure 4.4 shows the simulation result with and without applying adaptive particle sampling. The particle in the simulation result without applying adaptive particle sampling is rendered as the sphere as usual. Adaptive sampling does preserve the thread formed by the flow of melting liquid, comparing to the one without particle split.

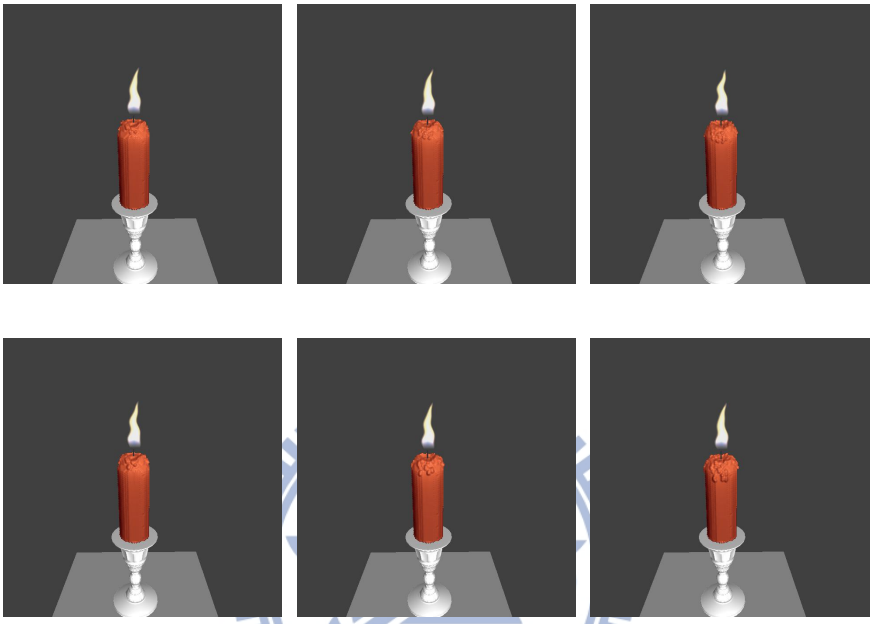
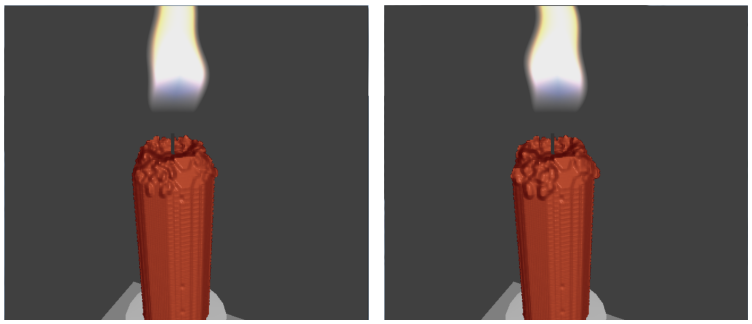


Figure 4.4: The upper row shows the simulation result without adaptive particle sampling. The droplet simply flows on the surface. The bottom row shows that the thread is preserved with particle splitting.

Figure 4.5 shows the comparison between rendering the particle as the sphere and the ellipsoid. Translucent effects are removed in order to clearly observe the sharp shape of melting droplets. The shape of small droplets after splitting is sharper in Figure 4.5(b).



(a) (b)

Figure 4.5: (a) Render the droplet using the sphere (b) Render the droplet using the ellipsoid.

Figure 4.6, 4.7, and 4.8 shows a comparison of a candle melting scene between our result, Carlson et al. [CMRBVHT02], and Pavia et al.[PPLT09]. Different from their melting behavior, our simulation model focuses on the surface flow of melting liquid and looks much more realistic.

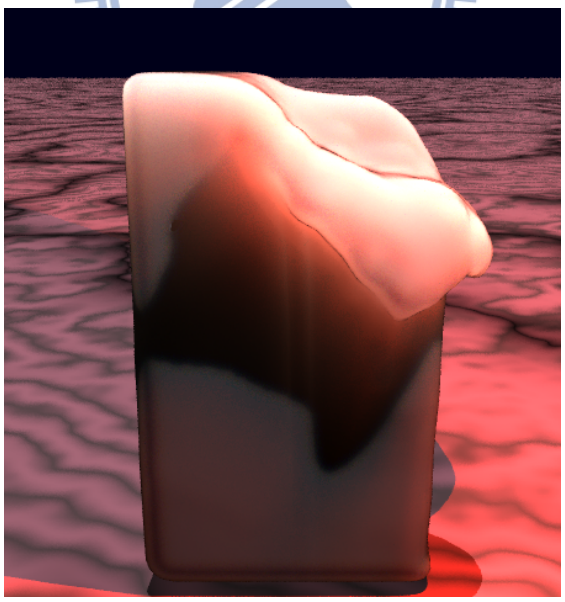


Figure 4.6: Wax melting in [CMRBVHT02].

Figure 4.7: Candle melting in [PPLT09].

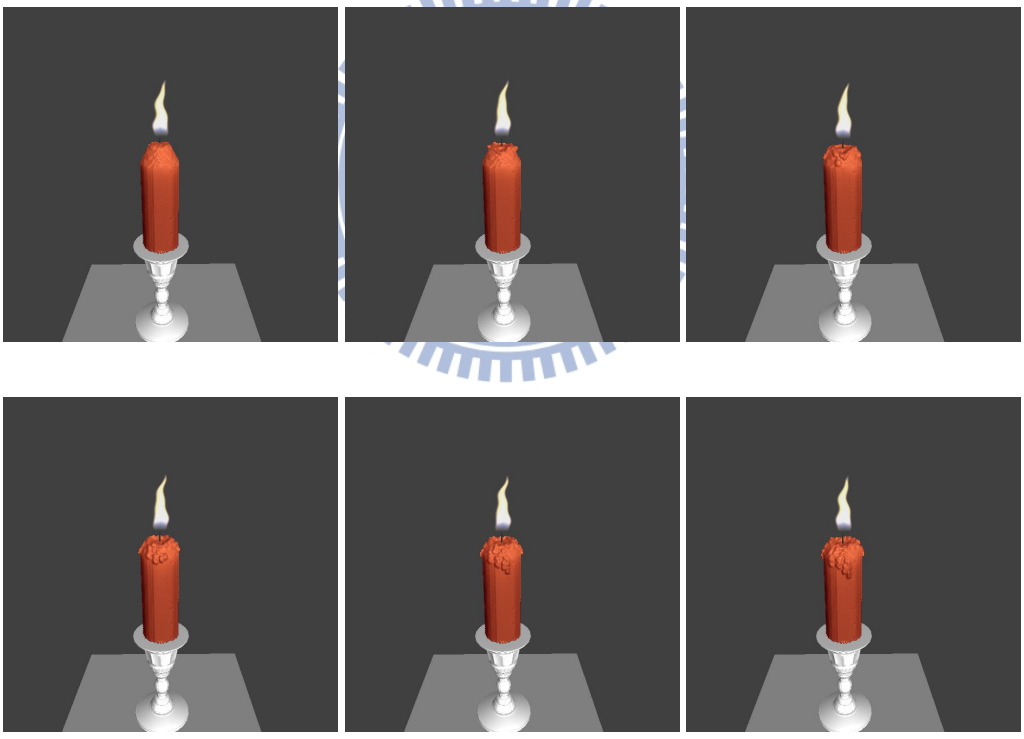


Figure 4.8: Our simulation result

CHAPTER 5

Conclusions

In this chapter, we briefly summarize our candle melting approach and rendering framework. Also, we describe the limitations and some future directions of improvement.

5.1 Summary

We have proposed a SPH-based simulation framework for candle melting. By utilizing the concepts proposed by Schechter et al. [SB12], we solve the density deficiency problem near the fluid boundary and achieve no-penetration and no-separataion solid-liquid boundary condition easily. The melting liquids thus could run along the surface closely without introducing any artificial interfacial tension. Moreover, by applying XSPH and setting appropriate ghost-solid velocities, the fluids can flow on the surface smoothly and achieve the expected stickiness easily. We propose a complete transfer system that considers the heat radiation, conduction, and dissipation. We model the heat source as the light source, and the surface particles that receive the radiation energy are determined by using shadow map. The radiation energy attenuates with the distance and

the attenuation ratio is formulated using a function like Gaussian distribution. The heat energy dissipating from the fluid surface is also considered. The net heat energy then introduces the change of the temperature and conducts to the entire model smoothly. We apply adaptive particle sampling to preserve the thread formed by the flow of viscous liquid. The split criteria are physically-based and thus do not introduce additional computation cost.

We use screen space point splatting approach to reconstruct the fluid surface. We render the sphere particle in different sizes based on its mass and stretch the sphere to the ellipsoid, which is aligned with the surface normal to model the shape of the flowing thread.

5.2 Limitations and Future Work

Although our adaptive particle sampling scheme could preserve the thread formed by the viscous liquid, its framework is incomplete and has several drawbacks. First, because the physics behind the formation of the thread may be complicated, the split criteria in our adaptive sampling scheme may not able to preserve the thread properly in some cases. Second, we lack particle merge operation and the simulation cost gets higher with the increase of sampled particles. Moreover, because we model the surface simply by a group of geometries (spheres or ellipsoids), aliasing appears on the fluid boundary due to sparse particle distribution.

In the future, the criteria of our split operation may additionally take the geometry features into consideration, such as extended local feature size (ELFS) proposed by Adams et al.[APKG07]. In addition, to use the computation resource more efficiently, we could add the particle merge operation to merge the particles inside the object. Finally, to get a smooth fluid boundary, we may increase the amount of particles involved in the simulation. However, it also increases the simulation cost.

Bibliography

[AIAT12] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. Parallel surface reconstruction for particle-based fluids. *Computer Graphics Forum*, 31(6):1797–1809, 2012.

[APKG07] B. Adams, M. Pauly, R. Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2007)*, 26(3), July 2007.

[ATT12] R. Ando, N. Thurey, and R. Tsuruno. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8), 2012.

[BT07] M. Becker and M. Teschner. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 209–217, 2007.

[CMRBVHT02] M. Carlson, P. J. Mucha, III R. B. Van Horn, and G. Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer Animation*, SCA '02, pages 167–174, 2002.

[CS08] H. Cords and O. Staadt. Instant liquids. In *Poster Proceedings of*

the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '08, 2008.

[DC96] M. Desbrun and M.-P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation, EGCAS '96, August, 1996*, pages 61–76. Springer, 1996. Published under the name Marie-Paule Gascuel.

[DS03] C. Dachsbacher and M. Stamminger. Translucent shadow maps. In *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03, pages 197–201, 2003.

[FM07] M. Fujisawa and Kenjiro T. Miura. Animation of ice melting phenomenon based on thermaldynamics with thermal radiation. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE '07, 2007.

[Gre10] S. Green. Screen space fluid rendering for games. In *Game Developers Conference*, 2010.

[HWC+05] C.-C. Ho, F.-C. Wu, B.-Y. Chen, Y.-Y. Chuang, and M. Ouhyoung. Cubical marching squares: Adaptive feature preserving surface extraction from volume data. *Computer Graphics Forum (Proceedings of Eurographics 2005)*, 24(3):537–575, 2005.

[IUDN10] K. Iwasaki, H. Uchida, Y. Dobashi, and T. Nishita. Fast particle-based visual simulation of ice melting. *Computer Graphics Forum*, 29(7):2215–2223, 2010.

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, 2003.

[MGG+10] N. Maréchal, E. Guérin, E. Galin, S. Mérillou, and N. Mérillou. Heat transfer simulation for modeling realistic winter sceneries. *Computer Graphics Forum*, pages 449–458, 2010.

[Mon05] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, August 2005.

[MSD07] M. Müller, S. Schirm, and S. Duthaler. Screen space meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 9–15, 2007.

[PPLT06] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. Particle-based non-newtonian fluid animation for melting objects. In *SIBGRAPI'06*, pages 78–85, 2006.

[PPLT09] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. Particle-based viscoplastic fluid/solid simulation. *Computer-Aided Design*, 41(4):306–314, April 2009.

[RWT11] K. Raveendran, C. Wojtan, and G. Turk. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 33–42, New York, NY, USA, 2011. ACM.

[SB12] H. Schechter and R. Bridson. Ghost SPH for animating water. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4), 2012.

[SG11] B. Solenthaler and M. Gross. Two-scale particle simulation. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4), 2011.

[SP09] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible SPH. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009)*, 28(3), 2009.

[SSP07] B. Solenthaler, J. Schläfli, and R. Pajarola. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds*, 18:69–82, 2007.

[vdLGS09] W. J. van der Laan, S. Green, and M. Sainz. Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D Graphics and Games*, I3D '09, pages 91–98, 2009.

[YT10] J. Yu and G. Turk. Reconstructong surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 217–225, 2010.