行政院國家科學委員會補助專題研究計畫成果報告

（□期中進度報告/■期末報告）

Melting Simulation and Flowing Control for Viscoelastic Models

高黏滯性物體之融化模擬與流動控制

計畫類別：■個別型計畫　□整合型計畫
計畫編號：NSC 100-2221-E-009-117-MY2
執行期間：100 年 8 月 1 日至 102 年 7 月 31 日

執行機構及系所：

計畫主持人：莊榮宏　　國立交通大學資訊工程系教授
共同主持人：
計畫參與人員：黃聰賢、姚博尹、官彥廷、黃富熙、張澤清

本計畫除繳交成果報告外，另含下列出國報告，共 _3_ 份：
■執行國際合作與移地研究心得報告
■出席國際學術會議心得報告

期末報告處理方式：
1. 公開方式：
   ■非列管計畫亦不具下列情形，立即公開查詢
   □涉及專利或其他智慧財產權，□一年□二年後可公開查詢
2. 「本研究」是否已有嚴重損及公共利益之發現：■否 □是
3. 「本報告」是否建議提供政府單位施政參考 ■否 □是，____（請列舉提供之單位；本會不經審議，依勾選逕予轉送）

中　華　民　國 103 年 1 月 22 日

Melting Simulation and Flowing Control for Viscoelastic Models
高黏滯性物體之融化模擬與流動控制

Abstract

The melting process of viscoelastic objects is a complicated phenomenon in real world, which includes fluid flowing, thermal radiation and heat transfer, and transformation between solid and fluid. We develop a SPH-based simulation framework for object melting simulation and a feedback control algorithm to manipulate the melting flow. By adding an elastic stress term to the Navier-Stokes equation, we can simulate the melting behavior which has both fluid and solid features. We develop a thermals transfer system to simulate the heat exchange between particles, the viscosity and elastic stress coefficient are automatically determined base on the temperature of particles. Moreover, we introduce a feedback control algorithm which can be used to intuitively design the flowing behavior. Users can draw flowing path directly on the surface to design the shape of melting fluid. This is a two-year project. In the first year, we develop and implement the SPH-based simulation system for object melting. In the second year, we develop a feedback control scheme that allows artists to design the melting flow and its final shape, and speed up the thermal exchange computation and develop a GPU-based computing framework.

Keywords

Malting simulation, Fluid control

摘要

高黏滯彈性物體之融化在真實世界中是一相當複雜的現象，包含融化物質之流動、熱量的傳遞轉移以及物質之相態變化等。我們利用平滑粒子流體動力學法 (Smoothed-Particle Hydrodynamics, SPH) 來模擬物體融化，並且使用一個具有回饋機制的控制演算法來操縱融化的流體。在納維 -斯托克斯方程式 (Navier-Stokes equation) 中加入計算彈性壓力的力量，讓我們可以模擬出同時具有固態與液態特性的高黏滯彈性流體。我們還在粒子間模擬熱能傳導的行為，利用模擬後的溫度決定黏滯性與彈力的模擬係數。此外，我們設計了一套直覺的控制機制來幫助使用者設計融化的過程。使用者可以直接在場景物表上繪製融化的路徑與融化後流體的形狀。本計畫為期兩年，第一年著力於高黏滯彈性物體融化模擬系統之發展與實作；第二年設計一流體控制系統，並發展方法加速熱量傳遞之計算以及 GPU 加速技術。

關鍵字

融化流體模擬、流體控制

Contents

1 Introduction (前言與研究目的)

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 project, we focuses on the simulation of 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 project, we develop a framework for melting and rendering. Also, we provide a straight-forward fluid control system, which allows animators to design the flowing pattern and the melting paths. 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.

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.

The contributions of the proposed 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.

- Provide an intuitive controlling framework, which allows user to design the melting flow and the final shape easily.

2 Related Work (文獻探討)

In this section, 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 1). Since then, the development of largrangian-based fluid simulation, especially using SPH, has made great progress.

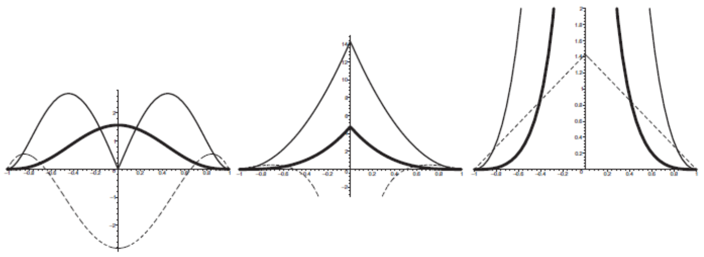


Figure 1: Three smoothing kernels W_{poly6}, W_{spiky} and $W_{viscosity}$ (from left to right). The thick lines are the kernels, the thin lines are the gradients in the direction towards the center, and the dash lines are the Laplacian [MCG03].

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.

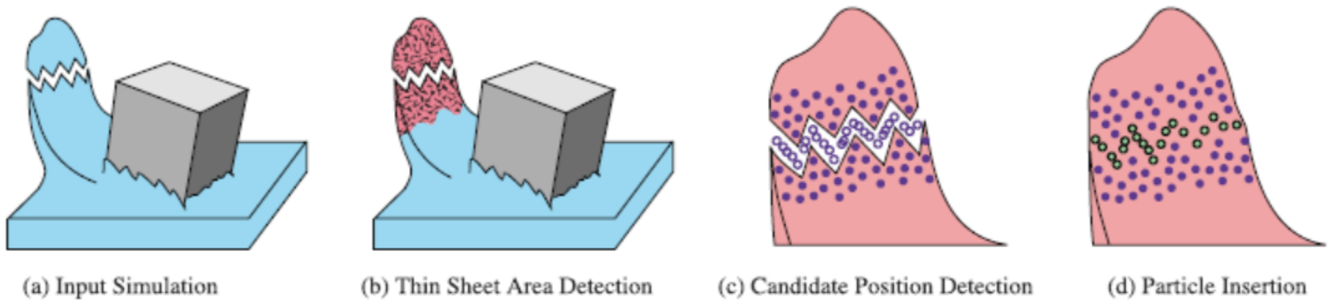

(a) Input Simulation
(b) Thin Sheet Area Detection
(c) Candidate Position Detection
(d) Particle Insertion

Figure 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 previous 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 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 4 demonstrates the heat transfer model.

2.3 Fluid Control System

Simulation of realistic fluid behavior has been always a hot field in computer graphics.Recently, more and more researchers have focused on the fluid control mechanism for artistic reasons in artist. Foster et al. [FF01] was the first to apply control scheme on fluids. They produced 3D parametric space curves to locally control the velocity and pressure of the fluid flow. Rasmussen et al. [REN+04] defined four types of control particles according to the liquid variables, including velocity, viscosity, level set, and the velocity divergence particles, where velocity particles controlled the movement of the liquid interface; viscosity particles for viscous melting behavior; level set particles is used to modified the level set representation in liquid interface; divergence particles modeled the liquid expansion and contraction effects. Treuille et al. [TMPS03] enforced the smoke to match the given velocity and density keyframes by building an optimization technique to solve the control parameters.

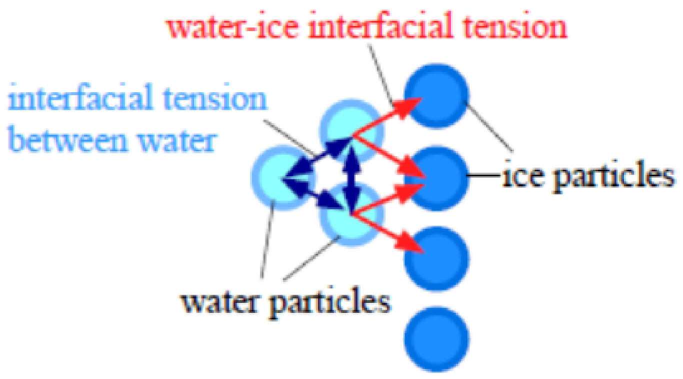

water-ice interfacial tension
interfacial tension between water
ice particles
water particles

Figure 3: Interfacial tension in [IUDN10].

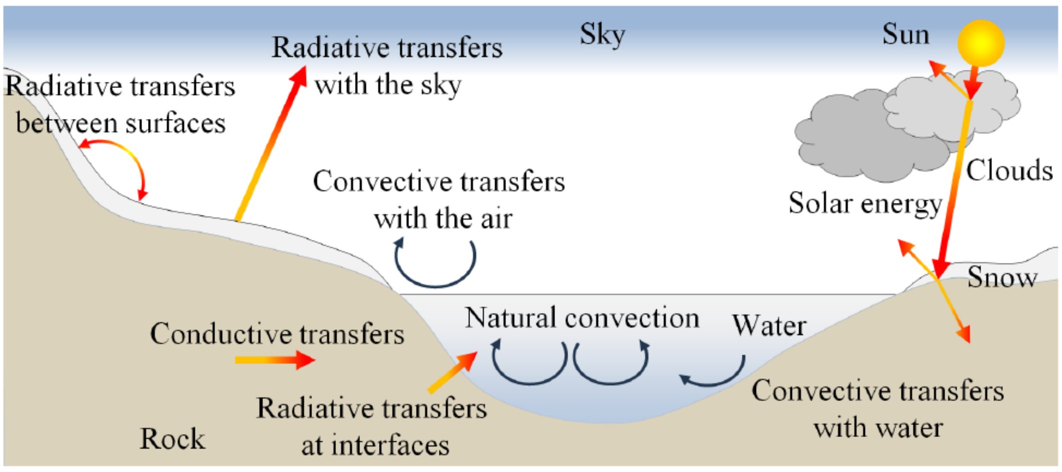

Sky
Sun
Radiative transfers with the sky
Radiative transfers between surfaces
Clouds
Convective transfers with the air
Solar energy
Snow
Conductive transfers
Natural convection
Water
Radiative transfers at interfaces
Convective transfers with water
Rock

Figure 4: Thermal transfer in [MGG$^+$10].

McNamara et al. [MTPS04] used the adjoint method to improve the efficiency of solving the nonlinear optimization problem. The key to their approach is to utilize the linear duality for variables substitution. The adjoint method exploited this powerful aspect of duality to accelerate the gradient calculation in optimization process. Fattal et al. [FL04] controlled the smoke shape with a sequence of predefined target smoke states. They added two terms to the standard flow equations : driving force term and smoke gathering term. While the driving force compelled the fluid to a particular smoke shape, the gathering term gathered the fluid to maintain the density value. Instead of solving the control forces optimization problem, Hong et al. [HK04] created a geometric potential field to exert the external force, and coerce the fluid to form the target shape. Shi et al. [SY05] controlled the liquids to match with the rapidly changing targets by building two different external force fields. The first one is a feedback force field applied to both shape and velocity to compensate the discrepancies with the target field. The second one is the gradient field of an adaptive geometric potential field defined by the target object shape and skeleton information. Compared to the original geometric potential field in [HK04], their adaptive potential field could handle different complex target shape with appropriate force magnitude. Later, Thürey et al. [TKPR06] introduced a control particles system to control the liquid without losing the detail part. As show in Figure 2.4, they first generated control particles set using some predefined criteria, then computed the attraction force and velocity force acting on the general fluid elements separately. In the end, they combined the control forces with weighting sum to evaluate the final forces for each element. In order to alleviate the virtual viscosity effect, they also applied high and low pass filter to preserve the detail motion. More recently, Dobashi et al. [DKNY08] developed a cloud formation control system. They provided a geometric potential field similar to the previous method [HK04] to solve the horizontal shape, but used a different feedback control scheme to control the cloud vertical height. Instead of feedback to the force strength, their feedback system controlled a water vapor supplier and a latent heat controller, allowing them to deal with the phase change effects and the cloud generation.

Though there are many fluid control methods, none of them had dealt with object melting and flowing problem, which is more complex because of the heat transfer involved in fluid elements and the melting behavior.

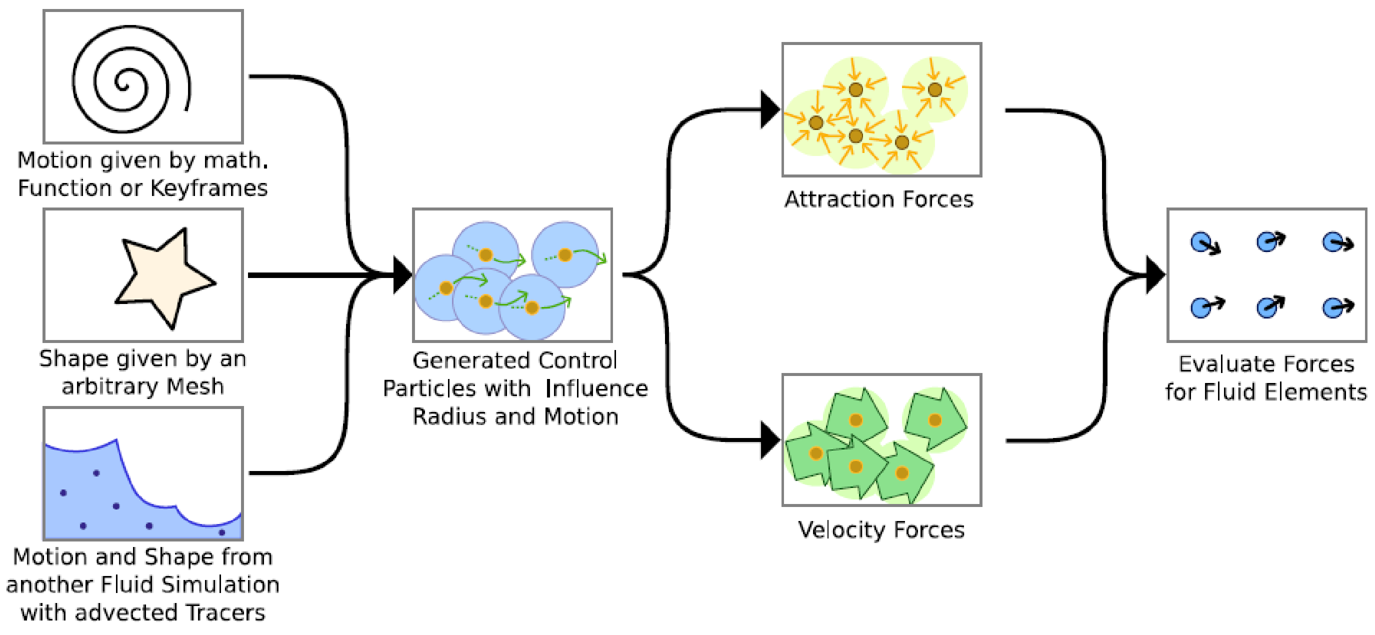

Motion given by math. Function or Keyframes
Shape given by an arbitrary Mesh
Motion and Shape from another Fluid Simulation with advected Tracers
Generated Control Particles with Influence Radius and Motion
Attraction Forces
Velocity Forces
Evaluate Forces for Fluid Elements

Figure 5: Particle-based control system [TKPR09].

2.4 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.

3 Melting Framework (研究方法)

3.1 Overview

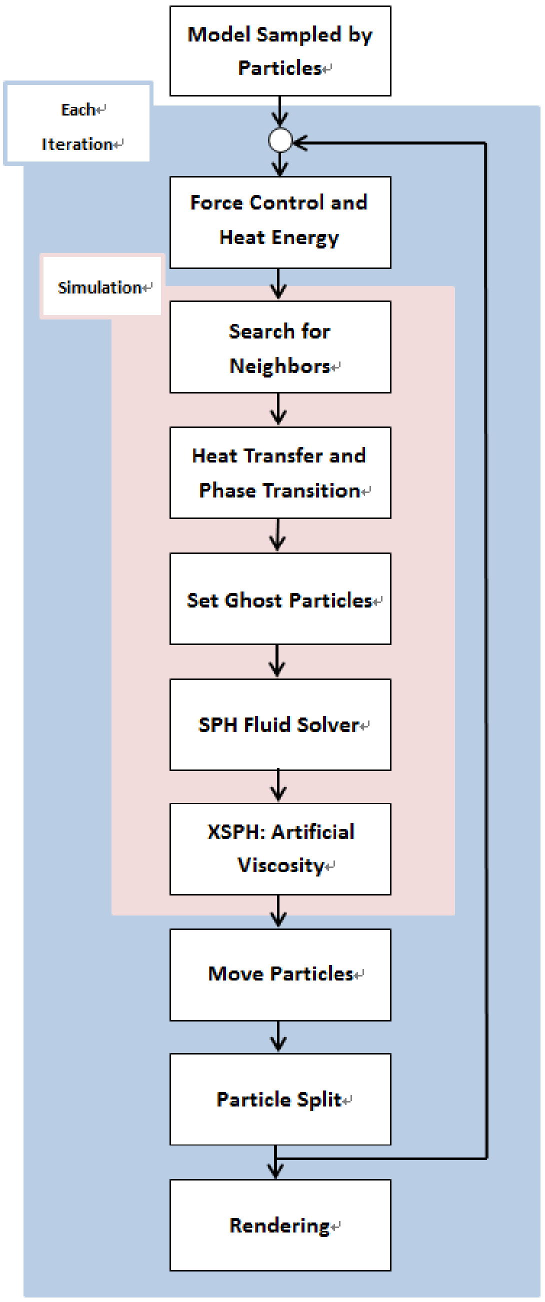


Figure 6: System flow of our algorithm

This project aims to naturally simulate the melting phenomenon, such as candle or butter. Figure 6 is the flow of our melting algorithm. First, giving a model as initial shape, 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 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 could 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. Then based on the output of the simulation part, we feedback to the control mechanism to adjust the force field automatically. Then, we can move the particles by an Euler scheme.

Finally, 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:

$$A(r) = \sum_j A(r_j)\frac{m_j}{\rho_j}W(\mathbf{r}-\mathbf{r_j}, h), \tag{1}$$

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 for each particle i by using

$$\rho(\mathbf{r}) = \sum_j m_j\frac{\rho_j}{\rho_j}W(\mathbf{r}-\mathbf{r_j}, h) = \sum_j m_j W(\mathbf{r}-\mathbf{r_j}, h), \tag{2}$$

which is also a derivation of Equation 1.

Besides, the gradient and Laplacian of A at location r could be derived easily by Equation 3 and 4

$$\nabla A(r) = \sum_j A_j\frac{m_j}{\rho_j}\nabla W(\mathbf{r}-\mathbf{r_j}, h) \tag{3}$$

$$\nabla^2 A(r) = \sum_j A_j\frac{m_j}{\rho_j}\nabla^2 W(\mathbf{r}-\mathbf{r_j}, h) \tag{4}$$

Fluids are often described by a velocity field **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

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0. \tag{5}$$

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

$$\rho(\frac{\partial\mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}) = -\nabla p + \mu\nabla^2\mathbf{v} + \rho\mathbf{g}, \tag{6}$$

where μ is the viscosity of the fluid, and **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,

$$\rho(\frac{\partial\mathbf{v}}{\partial t}) = -\nabla p + \mu\nabla^2\mathbf{v} + \rho\mathbf{g}. \tag{7}$$

There are three forces on the right hand side of the Equation 7 modeling pressure, viscosity, and external forces. We could formulate these force density terms acting on a particle i using SPH as follows [MCG03]:

$$\mathbf{f}_i^{pressure} = -\nabla p = -\sum_j m_j \frac{(p_i + p_j)}{2\rho_j} \nabla W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{8}$$

$$\mathbf{f}_i^{viscosity} = \mu \nabla^2 \mathbf{v} = \mu \sum_j m_j \frac{(\mathbf{v}_j - \mathbf{v}_i)}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{9}$$

To compute the pressure term in Equation 8 for each particle, we use ideal gas state equation

$$p_i = k(\rho_i - \rho_o), \tag{10}$$

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 11.

$$\mathbf{a}_i = \frac{\mathbf{f}_i}{\rho_i} = \frac{\mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscosity} + \rho_i \mathbf{g}}{\rho_i}, \tag{11}$$

where $\mathbf{a}_i$ is the acceleration of particle i. The velocity and the position of the particle i is then updated using Equation 12 and 13.

$$\mathbf{v}_i^* = \mathbf{v}_i + \mathbf{a}_i \Delta t, \tag{12}$$

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

$$\mathbf{r}_i^* = \mathbf{r}_i + \mathbf{v}_i^* \Delta t, \tag{13}$$

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 The fluid density distribution computed by Equation 2 is not even because the amount of neighbors between interior and boundary liquid particles differs. The pressure equation (Equation 10) leads to a case that particles near the boundary are clustered to a shell, aiming to rebalance the density; see the Figure 7 .

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 neighbor deficiency. Also, we add air ghost particles above the free surface of the fluid and within the kernel size [SB12]. 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 8.

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.

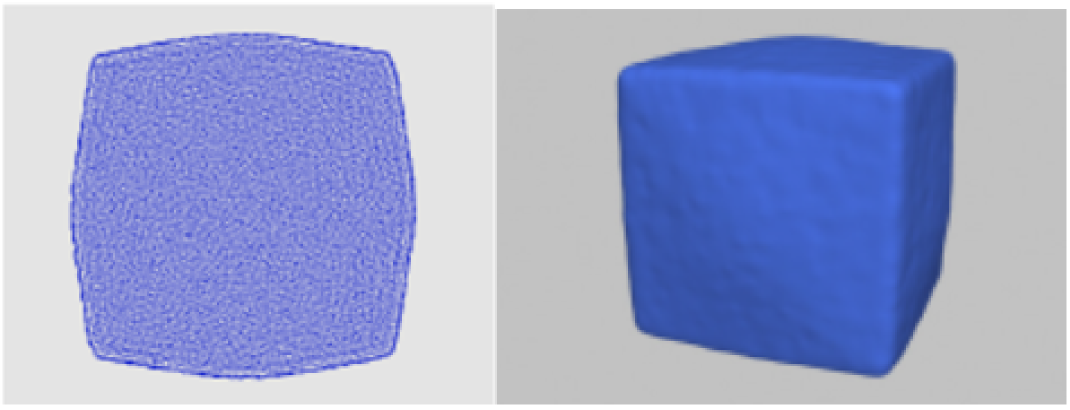


Figure 7: The fluid clusters into a shell due to neighbor deficiency [SB12].

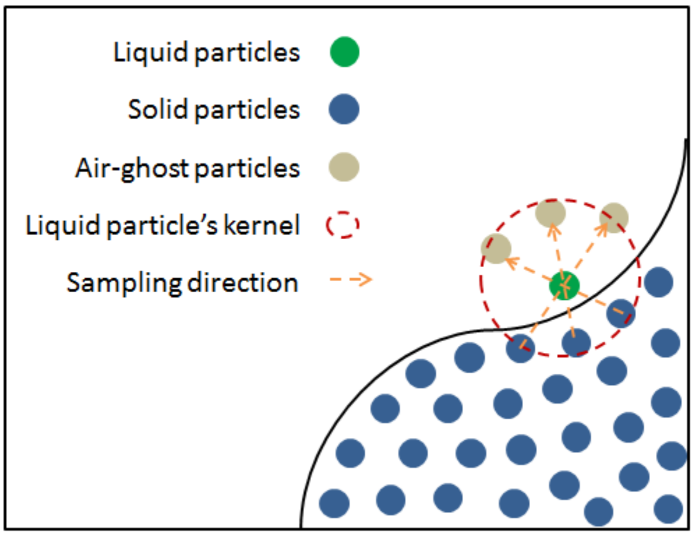


Figure 8: Sample air ghost particles.

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 14 :

$$\mathbf{v}_i^{new} = \mathbf{v}_i^* + \epsilon \sum_j \frac{m_j}{\rho_j} (\mathbf{v}_j^* - \mathbf{v}_i^*) W(\mathbf{r} - \mathbf{r_j}, h), \tag{14}$$

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].

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 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 15.

$$\mathbf{v}^{ghost} = \mathbf{v}_N^{solid} + \mathbf{v}_T^{liquid}. \tag{15}$$

By constructing the ghost velocity for solid particle as Equation 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.

For achieving the desired fluid stickiness more easily, we modify the Equation 15 as the following:

$$\mathbf{v}^{ghost} = \mathbf{v}_N^{solid} + \alpha \mathbf{v}_T^{liquid}, \tag{16}$$

where α, $0 \leq \alpha \leq 1$, is a tunable tangential friction coefficient. 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 9 shows the detail steps for computing the velocity of the melting liquids. We first compute the density for each liquid particle by using Equation 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 10 for each particle, we could compute the forces on the liquid particle by using Equation 8 and Equation 9. Then, we solve the velocity for melting liquids by Equation 12. Finally, we set the ghost velocity for solid particle using Equation 16 and smooth the velocity of liquid particles by applying XSPH (Equation 14).

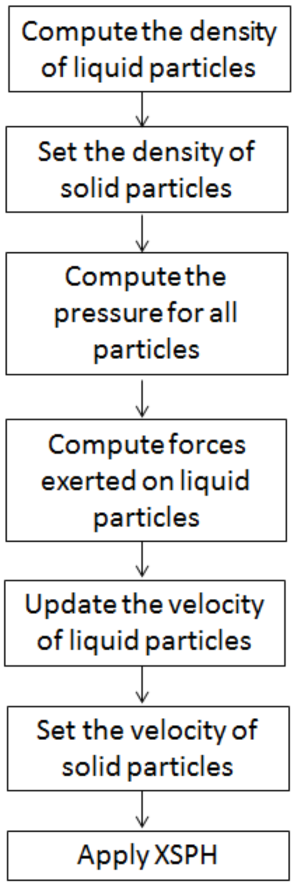

Compute the density of liquid particles
Set the density of solid particles
Compute the pressure for all particles
Compute forces exerted on liquid particles
Update the velocity of liquid particles
Set the velocity of solid particles
Apply XSPH

Figure 9: Flow of 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 10 shows each step of our heat transfer system. To determine the temperature of each 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

$$Q_i = Q_i^{radiation} - Q_i^{emission}, \tag{17}$$

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

$$\Delta T_i = \frac{Q_i}{m_i C}, \tag{18}$$

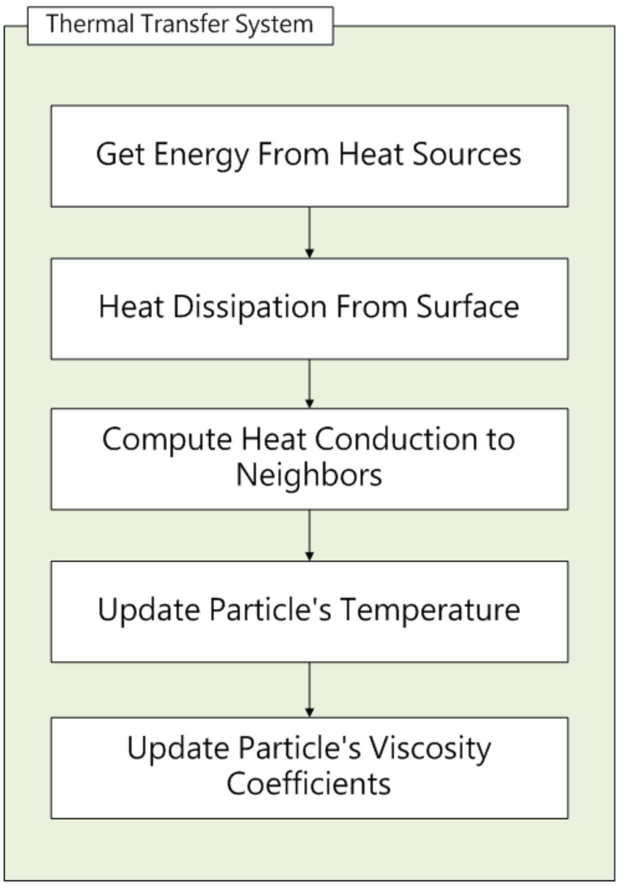

Thermal Transfer System
Get Energy From Heat Sources
Heat Dissipation From Surface
Compute Heat Conduction to Neighbors
Update Particle's Temperature
Update Particle's Viscosity Coefficients

Figure 10: Thermal transfer system flowchart.

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 9 for each liquid particle according to its temperature. Please see the Figure 11. The simulated viscosity force in Navier-Stokes equation (Equation 6) thus influences the motion of the melting liquids.

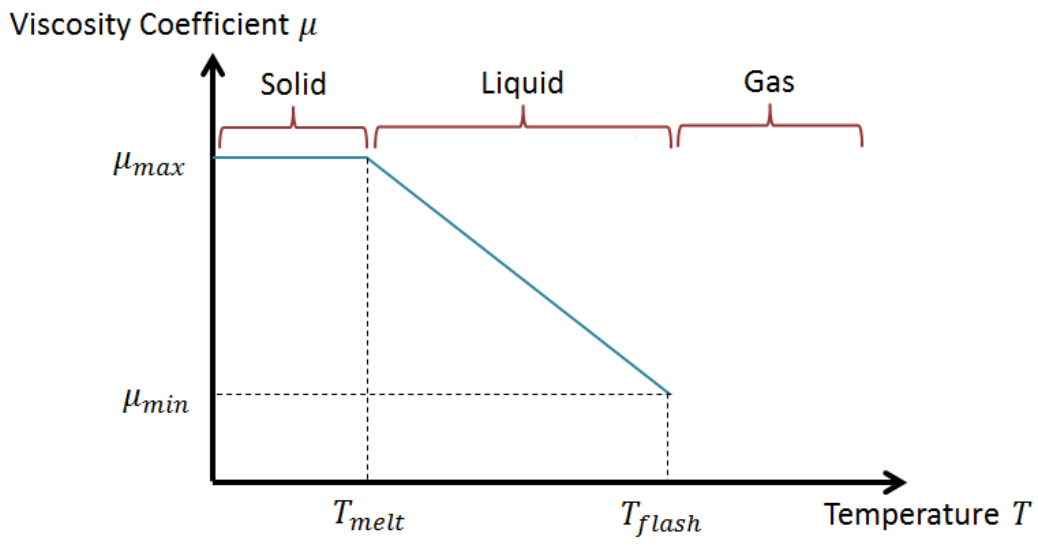


Figure 11: Interpolating viscosity coefficient.

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:

$$Q_i^{radiation} = \sum_s \delta Q_s \Delta t W(\mathbf{r} - \mathbf{p_s}, r_s), \tag{19}$$

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].

$$Q = \epsilon_e \sigma A(T^4 - T_{ambient}^4), \tag{20}$$

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.

$$Q_i^{emissive} = \epsilon_e \sigma 4\pi h_i^2 (T_i^4 - T_{ambient}^4), \tag{21}$$

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 18, the temperature conduction between particles could be calculated using the following equation:

$$\frac{\partial T_i}{\partial t} = c_d \sum_j m_j \frac{T_j}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h), \tag{22}$$

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 22 is a SPH-form of Equation 23 that gives the change in temperature T.

$$\frac{\partial T}{\partial t} = k\nabla^2 T - (\mathbf{v} \cdot \nabla)T, \tag{23}$$

where k is called the thermal diffusion constant, and **v** is the fluid velocity [CMRBVHT02]. The derivation from Equation 23 to Equation 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 22 could be derived by computing the Laplacian of the temperature $\nabla^2 T$ using Equation 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.

$$\frac{\partial T_i}{\partial t} = c_d \sum_j m_j \frac{(T_j - T_i)}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h). \tag{24}$$

Finally, the temperature after conduction is determined by Equation 25.

$$T_i^* = T_i + \Delta t \frac{\partial T_i}{\partial t}, \tag{25}$$

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

3.4 Adaptive Sampling

For the viscous liquid, it usually stretches to a thread when it flows, and we need to preserve this subtle shape. 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 26 and 27 after each split process.

$$m_i = \frac{m}{(s+1)^{l_i}}, \tag{26}$$

$$h_i = \frac{h}{\sqrt[3]{(s+1)^{l_i}}}, \tag{27}$$

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 12 demonstrates how we update the level of particles after split prcoess. Figure 13 shows the mass of particles computed by using Equation 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.

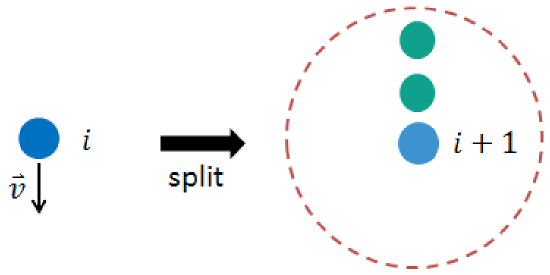


Figure 12: Case of s = 2. Splitting a particle of level i into three particles of level $i+1$. The newly sampled particles are placed behind the original one and within the kernel radius.

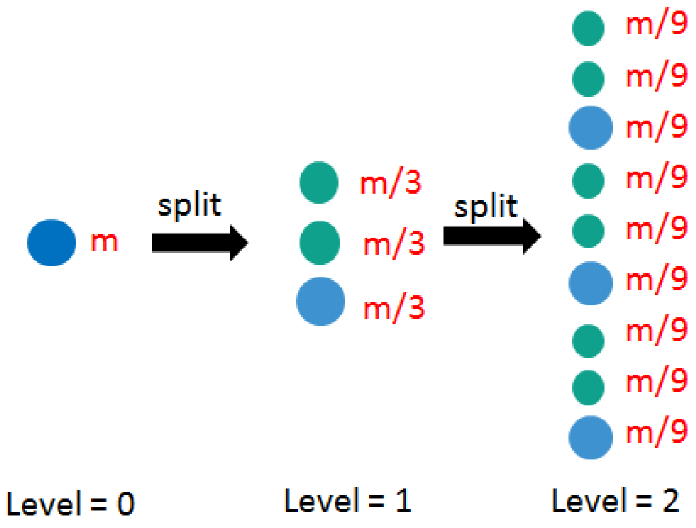


Figure 13: Case of s = 2. The change of mass for particles during the split process.

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].

$$\mathbf{f}_i^{pressure} = -\frac{m_i}{\rho_i}\sum_j \frac{m_j}{\rho_j}(p_i + p_j)\frac{(\nabla W(\mathbf{r_i} - \mathbf{r_j}, h_i) + \nabla W(\mathbf{r_i} - \mathbf{r_j}, h_j))}{2} \tag{28}$$

$$\mathbf{f}_i^{viscosity} = \mu\frac{m_i}{\rho_i}\sum_j \frac{m_j}{\rho_j}(\mathbf{v}_j - \mathbf{v}_i)\frac{(\nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h_i) + \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h_j))}{2} \tag{29}$$

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 14, 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.

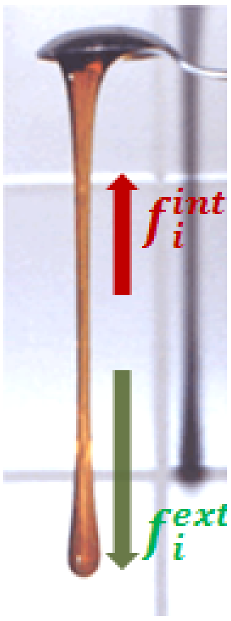


Figure 14: Real honey thread and the forces exerted on it.

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 Flowing Control System

In addition to simulating the object melting, the animators may expect to design the flowing pattern for object melting, including the shape of the fluid. Thus we develop a flowing control system, aiming to provide two types of control mechanism for different purpose. The first one is called flowing path control, which is used for guiding the direction of melting flow. The second one is named as melting shape control, which generate an approximate geometric potential field to constrain the flow shape. We utilize the control particles concept as in [TKPR09] to guide the fluid volume. Figure 15 shows the control system flowchart in our approach.

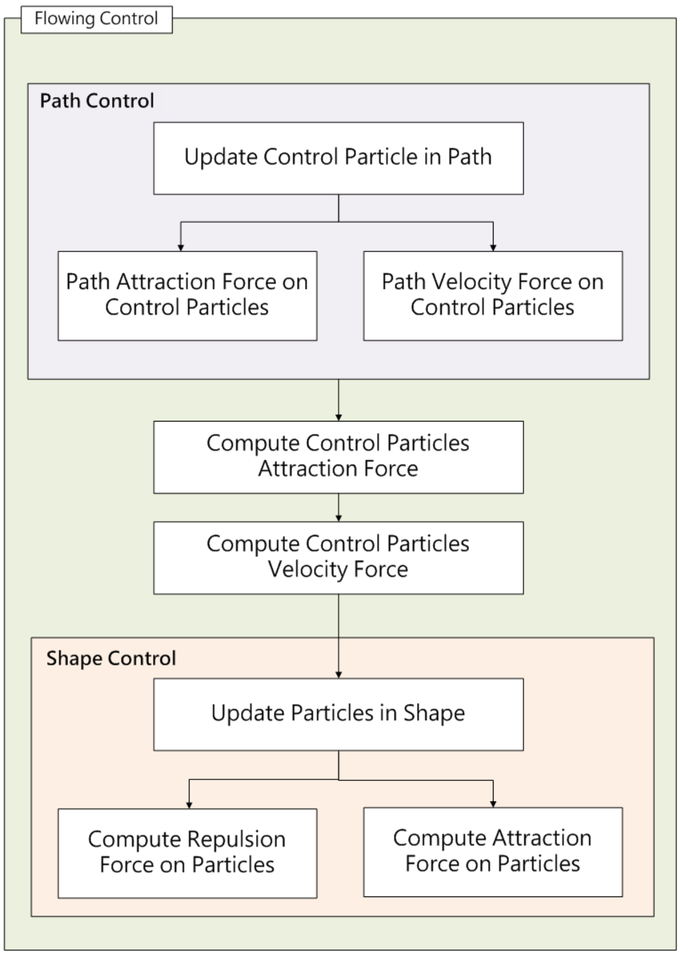

Flowing Control
Path Control
Update Control Particle in Path
Path Attraction Force on Control Particles
Path Velocity Force on Control Particles
Compute Control Particles Attraction Force
Compute Control Particles Velocity Force
Shape Control
Update Particles in Shape
Compute Repulsion Force on Particles
Compute Attraction Force on Particles

Figure 15: Flowing control flowchart.

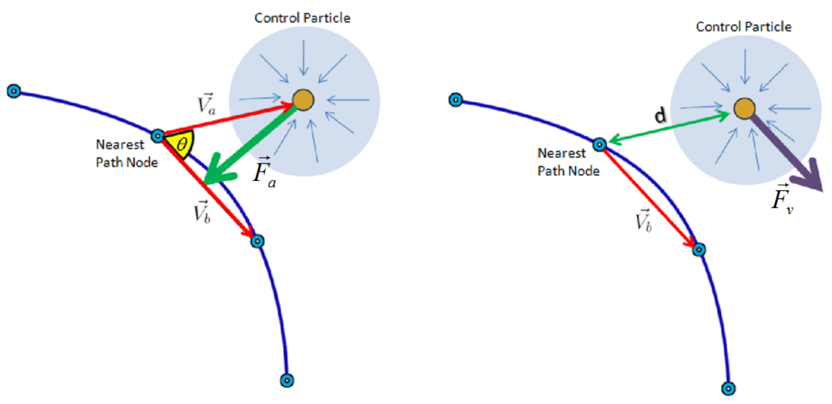
At the initial stage, we sample a set of control particles through the SPH particles. We define an influence range for each control particle to control the nearby particles. Each control particle has two forces to affect the movement of SPH particles. Attraction force drags the nearby particles to the control particles and the velocity force gives the nearby particles the forces based on the control particle's moving direction. The strength of force is inversely proportional to the distance between the control particle and other SPH particles. We make use of these control particles to guide all the SPH particles move as the user designed. Furthermore, we also utilize the attraction force to enhance small droplet formation during the melting process.

3.5.1 Path Control

For path control, user can draw several flowing paths directly on the object surface or other constrainted surface. These path are used for guiding the nearby control particles to flow along the path. Instead of apply forces on all the normal SPH particles, we consider the control particles, which is a subset of SPH particles, are enough to guide the fluid direction. Moreover, by lowering the strength of force with respect to the distance to the center of control particle, more fluid detail can be preserved. Each path is defined by a set of path nodes generated along a user-draw trajectory, the path nodes have the information about the influence range R, and two forces direction: attraction force and velocity force, which can be used to drag the control particles close to the path and give the control particles a direction along the path. In the end, each path may give the forces to control particles that will move along the path trajectory.

Attraction Force Our method compute the attraction force based on the vector V_a and V_b, which is shown in Figure 16(a). The control particle is attracted by the path along the direction $-V_a \sin\theta$, the angle θ is the angle between V_a and V_b. Equation 30 is the final attraction force for each control particle:

$$F_a = \sum_i (-\vec{V_a} \sin\theta)_i \tag{30}$$



(a) Attraction force (b) Velocity force

Figure 16: Two path control forces.

where i indicates the index of each path, F_a is the attraction force.

Velocity Force Velocity force provides the necessary force for fluid to flow along the predefined path. Figure 16 (b) represents how we generate the velocity force for each control particle. The path direction is decided by the vector from the current node to the next node position, and the magnitude of velocity force is proportional to the distance between the control particle and the path node. Equation 31 shows the formulation we use in our method:

$$F_v = \sum_i (\vec{V_b} W(d, R))_i \tag{31}$$

where i is the index of path, $\vec{V_b}$ is the direction along the path, $W(d, R)$ is a smoothing function (e.g. gaussian function) with influence range R defined on the path.

The final force for each control particle can be get as a weighted sum of F_a and F_v as shown in Equation 32.

$$F_{path} = (W_a * F_a + W_v * F_v) \tag{32}$$

In addition, we apply a feedback control scheme to adjust the force strength during simulation. If the fluid force direction is away from the control direction, we need a larger control force to correct the flowing direction. The strength of path control is proportional to the angle θ_p between the force direction provided by the fluid solver and the force direction provided by the paths. Equation 33 is the force provided by a path.

$$F_{Path_f} = W_p * F_{path} \tag{33}$$

Equation 34 shows the weighting coefficient W_p computation as a feedback scheme, where k_p is a constant coefficient and θ_p in Equation 35 is the angle between the force direction provided by the fluid solver and the force direction provided by the paths.

$$W_p = k_p(-\theta_p + 1.0) \tag{34}$$

$$\theta_p = \frac{F_{fluid} \cdot F_{path}}{\|F_{fluid}\| \|F_{path}\|} \tag{35}$$

where F_{fluid} is the force direction provided by the fluid solver and F_{path} is the force direction provided by the paths.

Except for the force control we have mentioned above, our path control also controls the temperature absorption rate to achieve our goal. Since the fluid will go toward low to flow, we slightly change the heat adoption coefficient δ at starting point of the path, so the melting may happen faster than other regions, causing the melting flow toward our control path. The heat adoption coefficient is simply modified by a Gaussian function, making the adoption coefficient smoothly varying according to the distance to the center of path point. Figure 17 shows the adoption coefficients varies in a color-coded particles diagram.

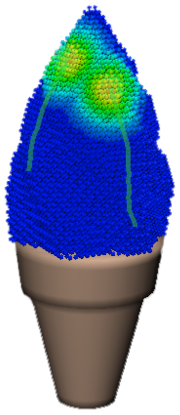


Figure 17: Modify the adoption coefficient around the path starting point

3.5.2 Shape Control

It is quite often that artists have the needs to control the fluid to form a target shape. In our object melting simulation, we provide an easy-design shape control interface for artists to do these jobs. Differ from path control scheme, shape control must control all the SPH particles more precisely to obey the constraints we set. Because user will not expect there are small droplet cross the shape or there are some details do not follow this constraint. Therefore, instead of only applying on control particles, we enforce all the SPH particles in the influence range of control shape to accept the control forces.

As the same way in path creation, we allow user to draw shape contours on the scene surface, and store each contours with discrete nodes along the trajectory. Although the shape contour is 2D on the surface, the influence is still on a 3D volume around each node. We think this is adequate for a melting process control since the fluid is usually flowing close to the surface. A shape contour has several shape nodes on the trajectory. And for each shape node, it stores a direction for the contour repulsion force and two influence range as shown in Figure 19. The farther range is the attraction range used to attract SPH particles to move close the contour line and the closer range is the repulsion range used to keep the particles stay inside the shape contour region.

Once a shape contour is drawn, we then compute the shape force direction for each node on the shape contour. By finding a nearest surface vertex normal vector and the vector to its next node position, we compute the shape force vector as the cross product of these two vectors, as shown in Figure 18.

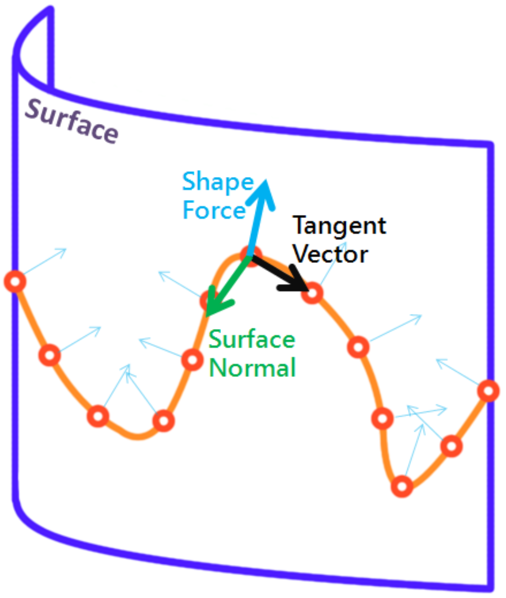


Figure 18: Shape control force for each node.

We divide the influence range for each node into two level: attraction and repulsion range. Figure 19 reveals the two level range in our approach. Inside the light red region, we employ the shape force describe above to enforce the fluid inside the shape contour. In the light blue region, we use a attraction force to attract the nearby particles to form our target shape, and the direction is simply the vector from particle to shape node.

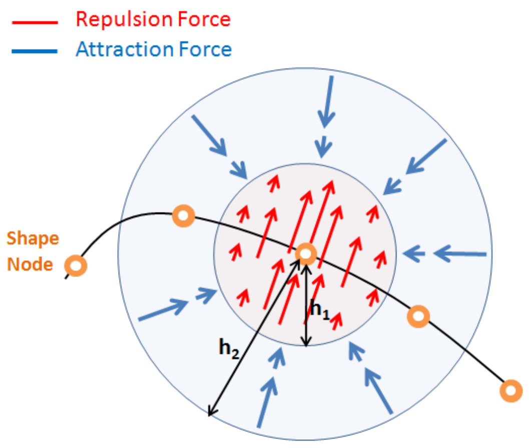


Figure 19: Shape node two level influence range

Equation 36 shows the formulation of shape attraction force, which is used to attract the nearby SPH particles to form the target shape.

$$F_{Sa} = C_{Sa} V_{Sa} W(d - h_1, h_2 - h_1) \tag{36}$$

where F_{Sa} is the attraction force, C_{Sa} is the coefficient of attraction strength, V_{Sa} is the attraction direction from particle position to the shape node position, and W is a smoothing kernel to smooth interpolate the magnitude of the force with the distance d.

The repulsion force computation can be done as Equation 37:

$$F_{Sr} = C_{Sr} V_{Sr} W(d, h_1) \tag{37}$$

where F_{Sr} is the repulsion force, C_{Sr} is the coefficient of repulsion strength, V_{Sr} is the repulsion direction for the shape node, and W is a smoothing kernel to smooth interpolate the magnitude of the force with the distance d.

Moreover, we also control the temperature of particles within the influence region. We slightly lower down the temperature per iteration until it is below to the melting point, so the particle will stick to the target shape we design.

3.6 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.

3.6.1 Screen Space Rendering

As shown in Figure 20, 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 21 shows the depth map with and without applying Gaussian filter.

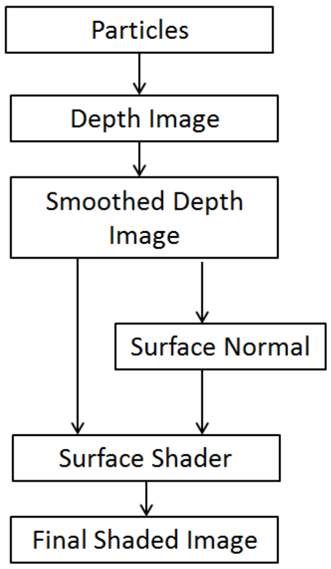


Figure 20: Flowchart of screen Space Rendering.

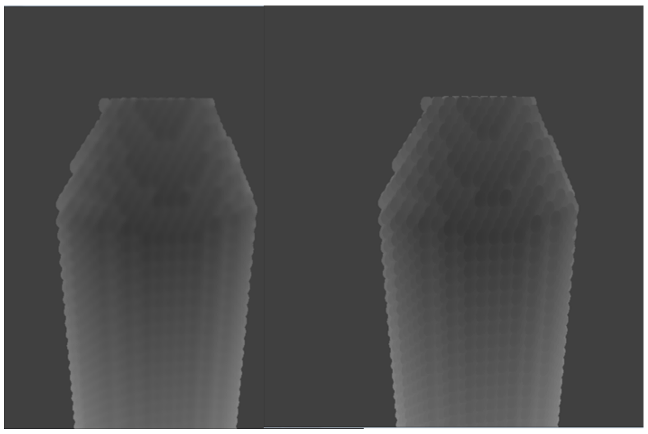


Figure 21: Depth map with/without Gaussian Filter.

After smoothing the depth map, we use it to compute the fluid normal vectors in per-pixel manner; see Figure 22. 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.

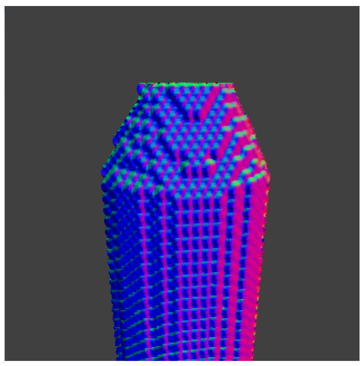


Figure 22: Normal map obtained from the depth map filtered with Gaussian filter.

3.6.2 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 23 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.

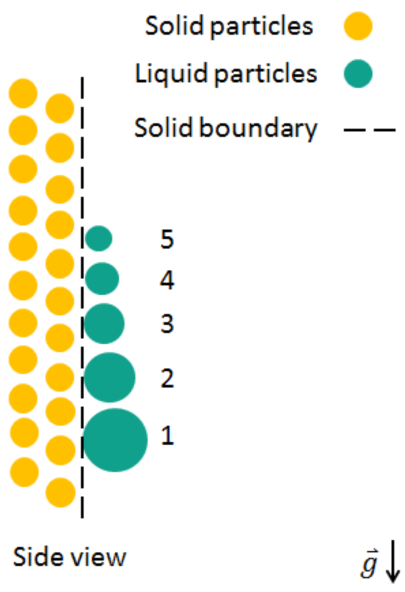


Figure 23: Render particles using spheres in different radius. The number indicates the particle's split level.

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 24.

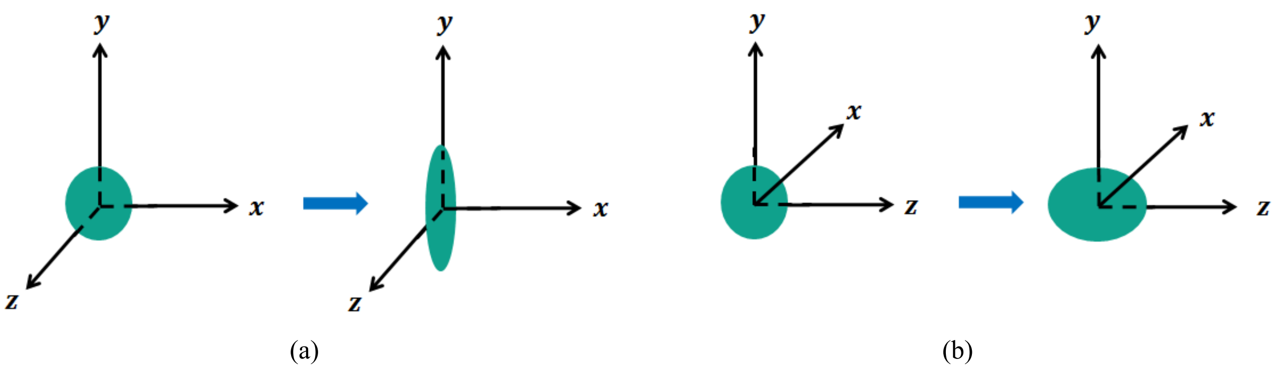


Figure 24: (a) Stretch the sphere in y-direction so that it looks like a filament. (a) Stretch the sphere in z-direction to make a sharp prominence.

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

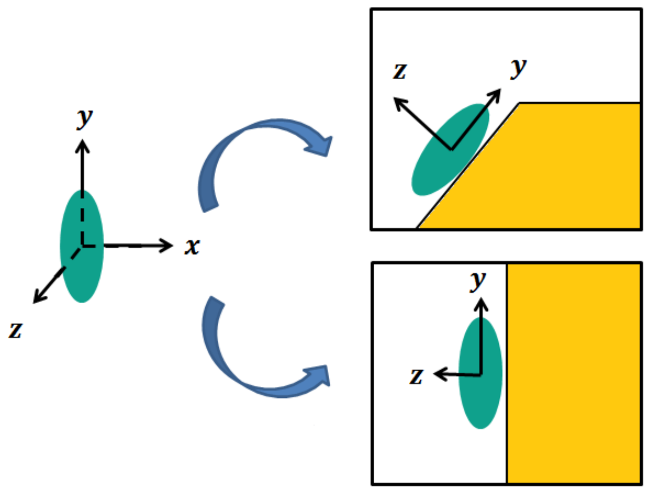


Figure 25: Rotate the ellipsoid to align the surface normal vector.

3.6.3 Translucent Rendering

Because the wax is a translucent material, we integrate Translucent Shadow Map [DS03] in our rendering framework. Figure 26 shows the standard process of Translucent Shadow Map.

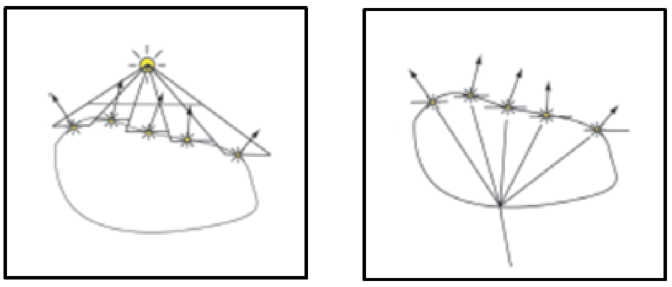


Figure 26: The Translucent Shadow Map stores irradiance samples (left). The radiance leaving the object is them computed by filtering these radiance samples (right) [DS03].

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 more stereoscopic.

4 Results (研究成果)

Here we show some results. First, the flow behavior can be adjusted by our parameter. Next, we display our method can preserve the details such as the thread of wax. Finally, we demonstrate our control ability.

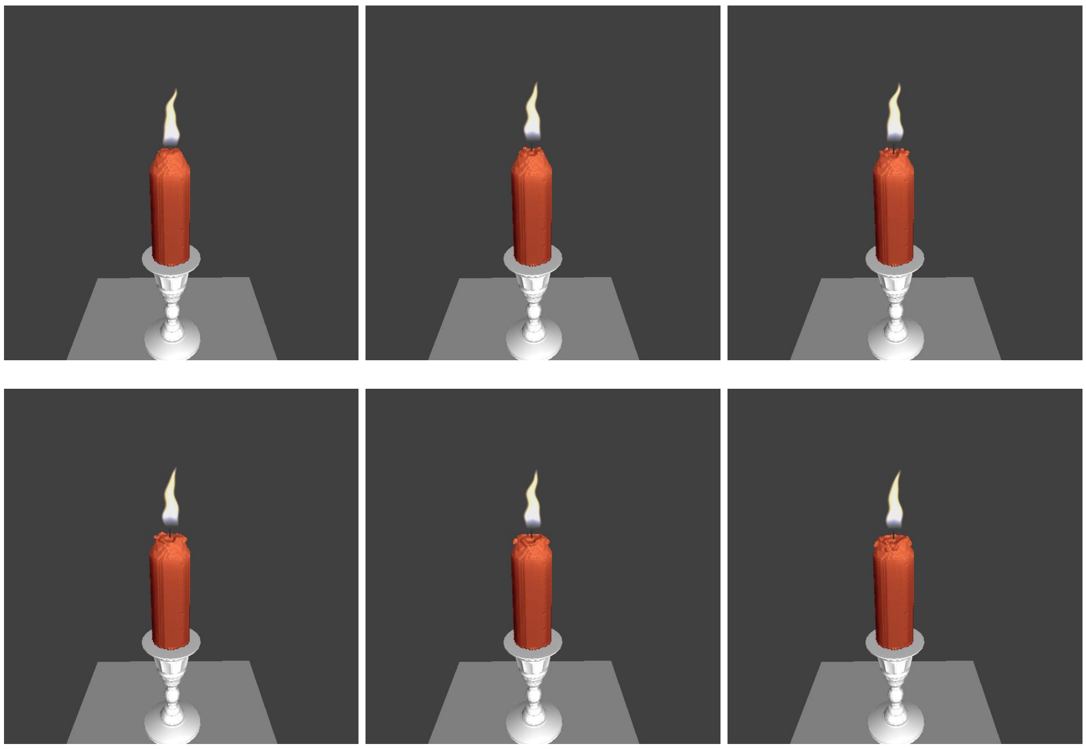


Figure 27: $\alpha = 0.2$. The fluid move slowly and almost stick on the solid surface.

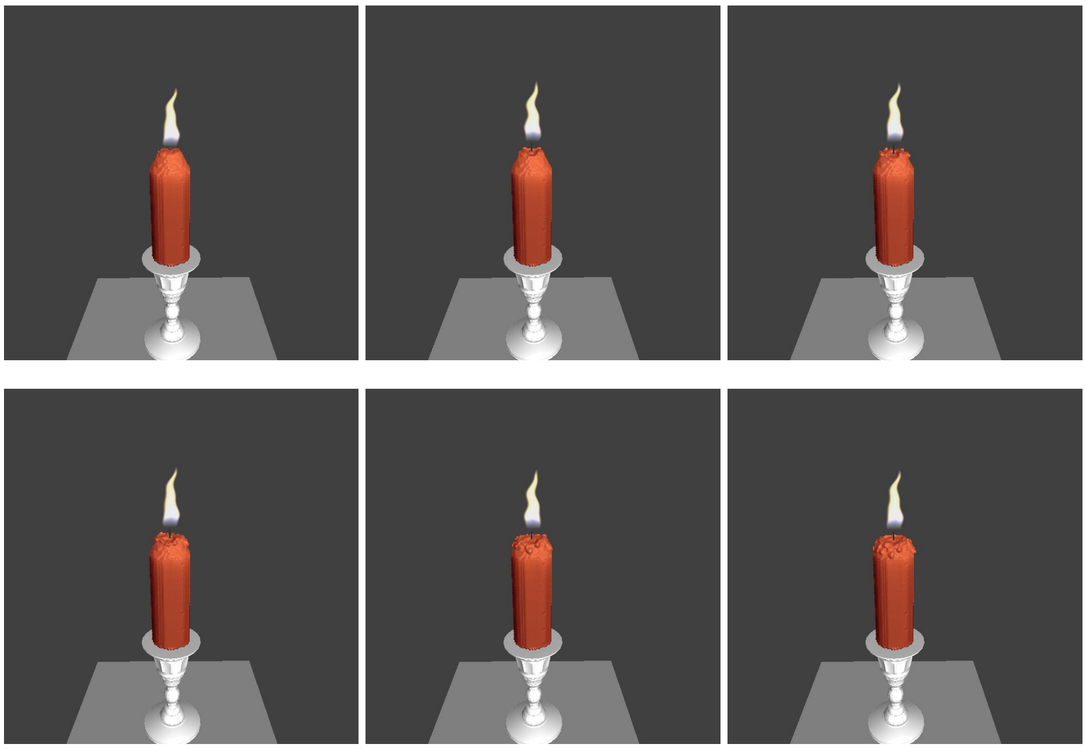


Figure 28: $\alpha = 0.5$. The fluid run along the solid surface smoothly.

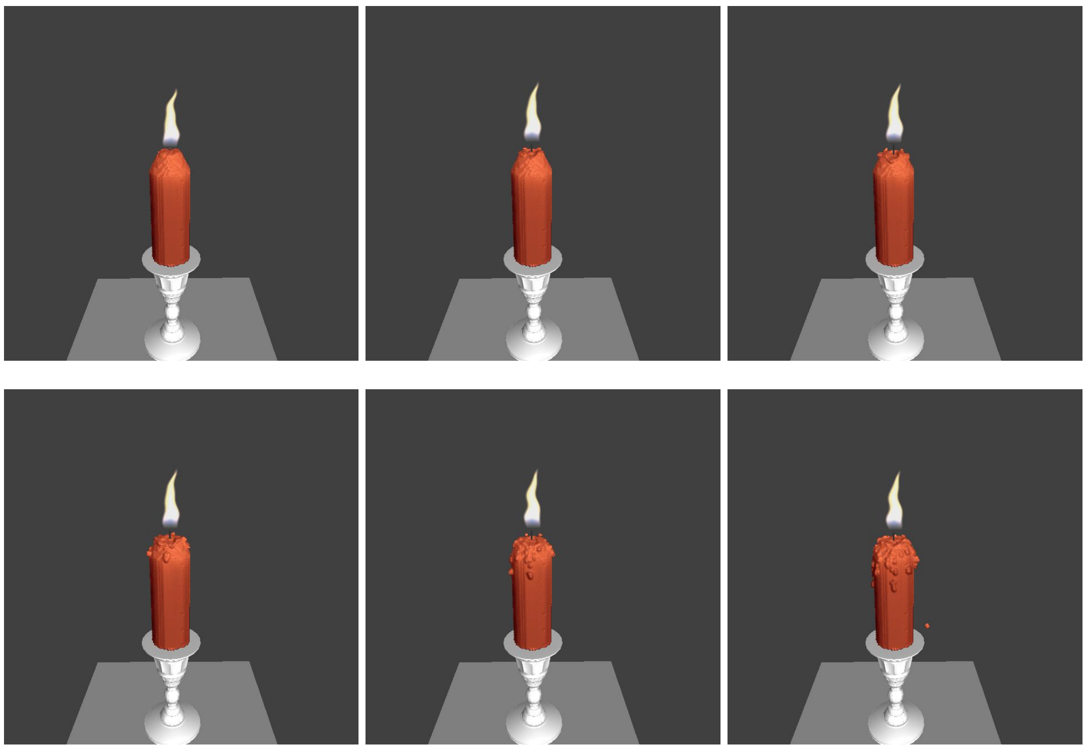


Figure 29: $\alpha = 0.8$. The fluid on the slope run away from the solid surface easily.

To illustrate that the parameter α appeared in Equation 16 determines the stickiness of the fluid, Figure 27, Figure 28, and Figure 29 show the simulation process with different α values. 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.

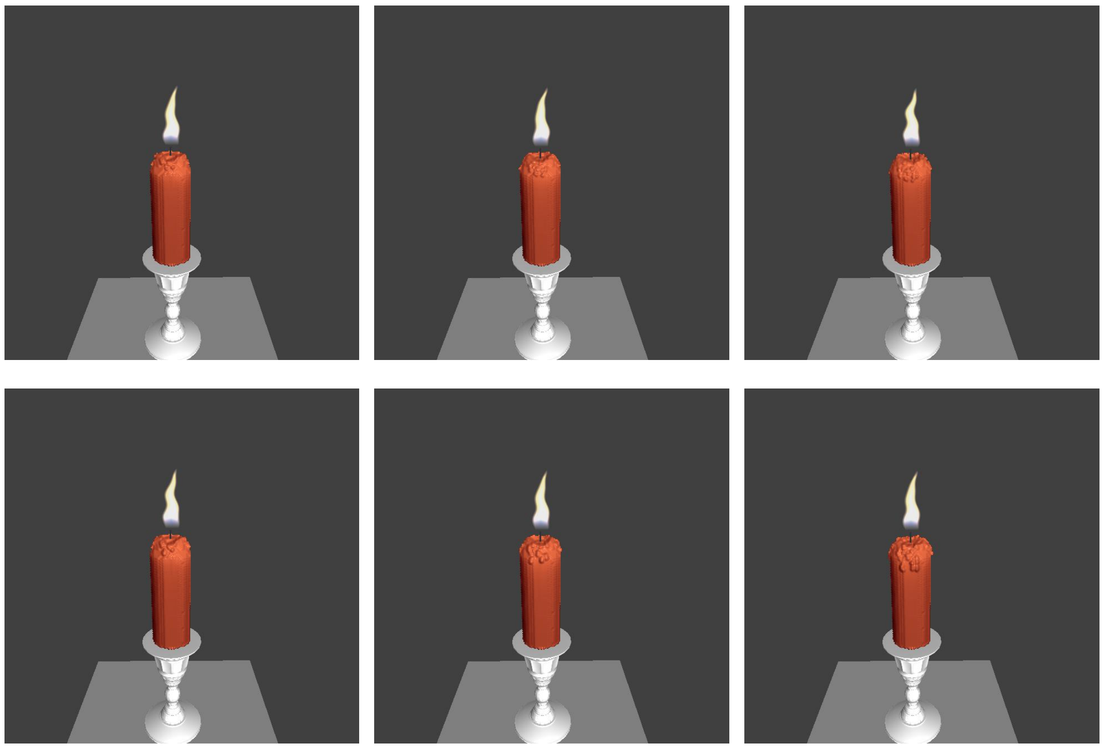


Figure 30: The upper row shows the simulation result without adaptive particle sampling. The droplet simply flows on the surface. The bottom row shows that the thread is preserved with particle splitting.

Figure 30 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.

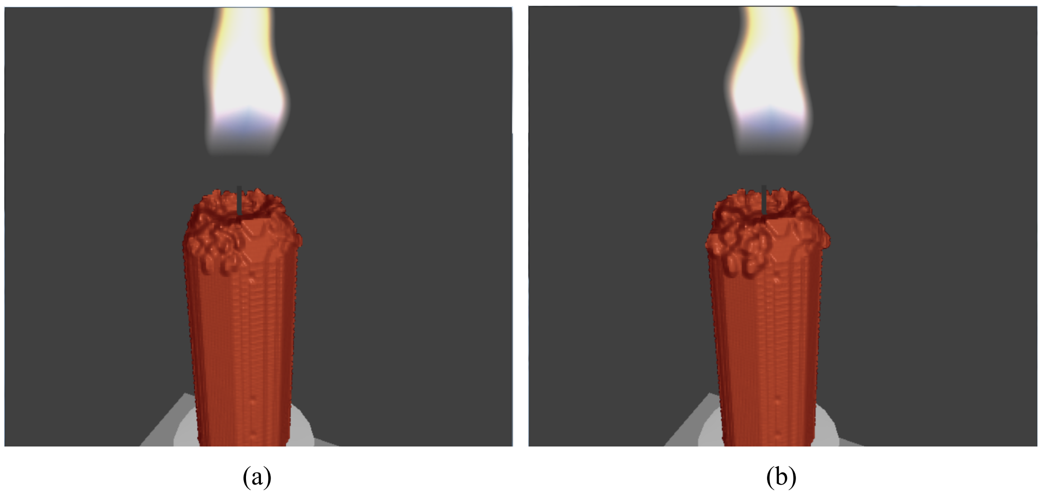


Figure 31: (a) Render the droplet using the sphere (b) Render the droplet using the ellipsoid.

Figure 31 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 Figure31(b).

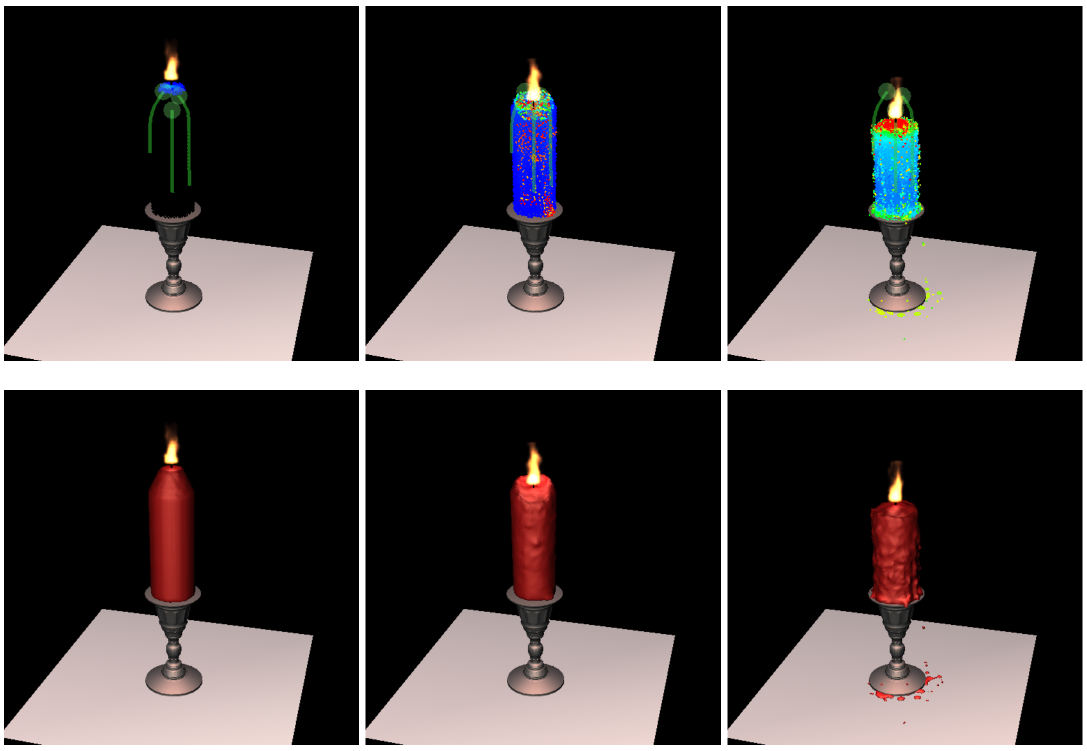


Figure 32: Candle melting with path control

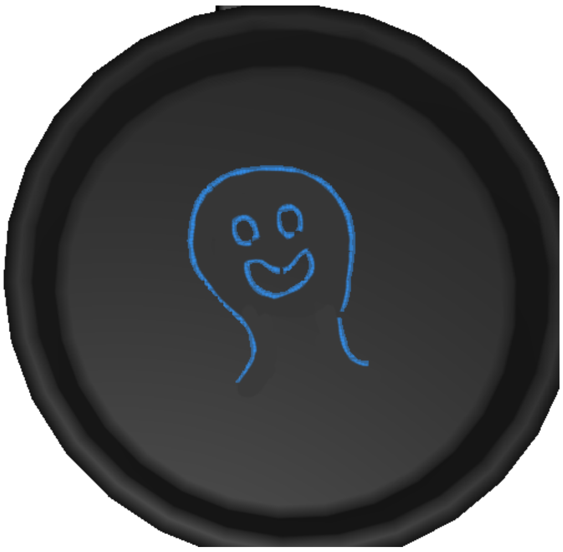


Figure 33: The shape control line drawn by user

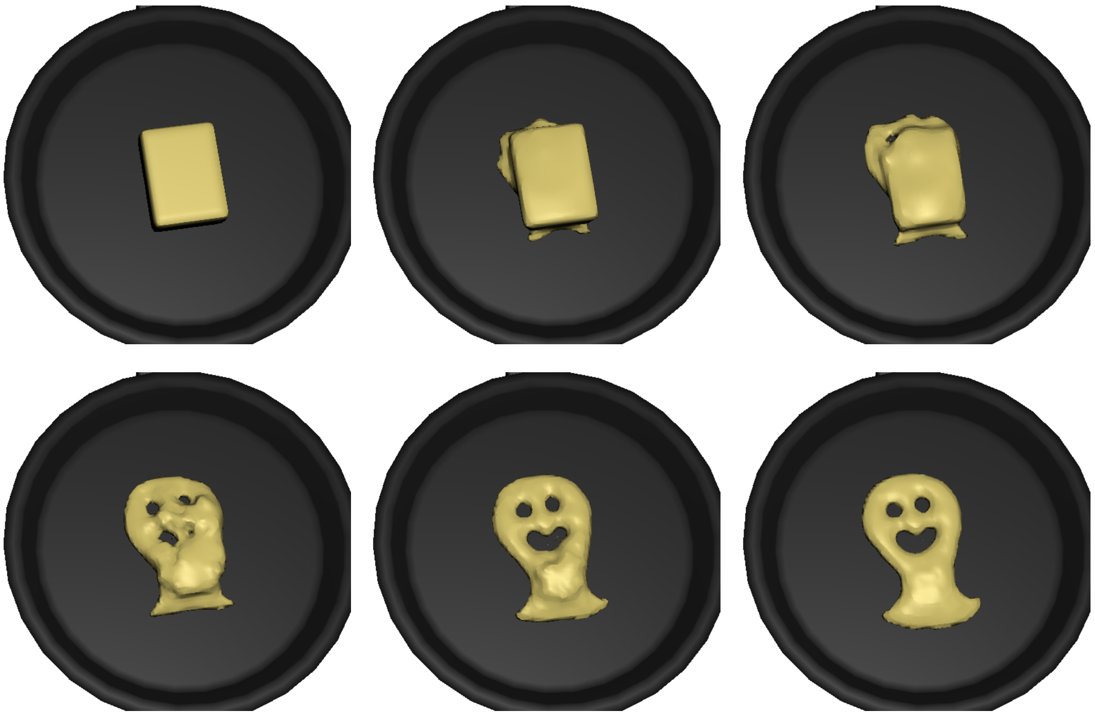


Figure 34: Butter melting with shape control

Figure 32 shows the user can draw the green path line on the melting object to guide the thread of the candle. Figure 33 show the desired input which is drawn by user. And Figure 34 displays the simulation result with shape control.

CPU	Intel core2 duo E6750 2.66GHz
GPU	Nvidia GTX260
OS	Windows 7
Programming language	C++ and OpenGL

Table 1: The information of simulation platform.

Speed up by CUDA	Particle Numbers	FPS
NO	29162	0.067
Yes	29162	6.649
Yes	16427	11.473

Table 2: The times of results with/out speeding up by CUDA

The simulation platform is shown in Table 1. Table 2 shows the times of simulation results with or without control. To use the graphics hardware, we use the NVIDIA CUDA to perform the parallel computing.

5 Conclusions (結論與成果自評)

The goal of this two-years project has been achieved. In the first year, we propose a system which integrates the SPH-based simulation, heat conduction, and adaptive merging or splitting the particles. 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 heat energy dissipates 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 also presented our result in the conference of Computer Graphics Workshop in 2012.

In the second year, we improve the system with flow control and GPU acceleration, and the quality of detail are also improved. The path control and shape control are achieved by adding attraction force. With the power of graphic hardware, the rendering speed gets significant improvement. 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. 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. To further improve the quality, we apply adaptive particle sampling to preserve the thread formed by the flow of viscous liquid. We also use screen space point splatting approach rather than marching cube to reconstruct the fluid surface. The particles are rendered in sphere with different sizes based on its mass. Then the particles are stretched as ellipsoids which are aligned with the surface normal to model the shape of the flowing thread. The entire computation can be performed on GPU to achieve a real-time rendering speed.

References

[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.

[TKPR09] Nils Thürey, Richard Keiser, Mark Pauly, and Ulrich Rüde. Detail-preserving fluid control. *Graphical Models*, 71(6):221--228, 2009.

[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.