Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

Fast Particle-based Visual Simulation of Ice Melting

K. Iwasaki[1], H. Uchida[1,2], Y. Dobashi[3], and T. Nishita[4]

[1] Wakayama University, Japan [2] Sony Computer Entertainment, Japan
[3] Hokkaido University, Japan [4] The University of Tokyo, Japan

Abstract

The visual simulation of natural phenomena has been widely studied. Although several methods have been proposed to simulate melting, the flows of meltwater drops on the surfaces of objects are not taken into account. In this paper, we propose a particle-based method for the simulation of the melting and freezing of ice objects and the interactions between ice and fluids. To simulate the flow of meltwater on ice and the formation of water droplets, a simple interfacial tension is proposed, which can be easily incorporated into common particle-based simulation methods such as Smoothed Particle Hydrodynamics. The computations of heat transfer, the phase transition between ice and water, the interactions between ice and fluids, and the separation of ice due to melting are further accelerated by implementing our method using CUDA. We demonstrate our simulation and rendering method for depicting melting ice at interactive frame-rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Visual simulations of natural phenomena, such as smoke, water, and fire, have been widely studied in the field of computer graphics, and have been used for many applications such as movies, commercial films, and games. Among many natural phenomena, ice is often depicted in movies and commercial films. Examples include icicles that are created by the freezing of meltwater under a roof in winter scenes and ice-cubes in a glass.

In this paper, we focus on the simulation of melting ice, and interactions between ice and meltwater. In the melting ice phenomena such as melting icicles, the meltwater forms a thin film of water that surrounds the ice and flows on the ice surface. The meltwater then forms a water droplet at the bottom of the ice, and the water droplet falls due to gravity. Our purpose is to develop a method that can simulate all of these phenomena.

Although several methods [CMIT02, LIGF06, SSP07] have been proposed to simulate melting, the flows of meltwater droplets on the surfaces of the objects are not taken into account. The common methods for the simulation of melting can be categorized into two techniques: grid-based approaches and particle-based approaches. The grid-based approaches subdivide the simulation space into a grid and calculate the physical properties at each grid cell. Although a grid-based approach can simulate realistic animations of fluids and solids, this approach is not very suitable for simulating ice melting because of the following reasons. Firstly, a high resolution grid would be required in order to represent the thin meltwater film and the water droplets. Secondly, a loss of fluid volume often occurs due to numerical diffusion. This makes it difficult to simulate the thin meltwater film and the water droplets, since the volumes of these are small and even a small loss of volume severely affects the simulation. Although a variety of techniques (e.g. the particle level set method) have been developed to improve the accuracy, the simulation time also then increases.

To simulate the flows of meltwater on an ice object efficiently, our method employs a particle-based approach. The heat transfer between the ice and other materials (e.g. water, air and heat sources), the phase transitions from ice to water, and vice versa, and the interactions between the meltwater and the ice can be simulated by using our particle-based approach. To simulate the thin water film around the ice and the meltwater droplets, the ice object is represented by densely-sampled particles. This results in an increase in the simulation time. To address this problem, we propose a GPU-based

simulation method for the melting of ice and the interactions between meltwater and ice.

The motion of the ice, represented by a set of ice particles, can be simulated in the same way as would be used for a particle-based rigid body simulation. That is, the force exerted on the ice is calculated by summing the forces working on all of the ice particles that belong to the entire mass of ice. However, the ice mass can separate due to melting, and therefore each ice particle has to be tracked taking into account whichever ice mass each ice particle belongs to. To solve this problem, we propose an efficient parallel algorithm and its GPU implementation.

The contributions of the proposed method are as follows:

- a particle-based simulation method is proposed for the heat transfer and interactions between meltwater and ice.
- a simple simulation method for the interfacial forces between water-ice and water-water particles is presented to represent the flows on the ice and the water droplets.
- the simulation, including separation of the ice, is accelerated by using GPUs.

2. Previous Work

Our purpose is to simulate the melting of ice, taking into account the flows of meltwater on the ice and the formation of water droplets. This section reviews previous methods for the simulation of melting and freezing phenomena, and also of water droplets.

Melting and Freezing Simulations : Fujishiro et al. [FA01] simulated the thawing of ice by using the morphology and cellular automaton. This method represents an ice object by using voxels and calculates the heat conduction and the melting effects based on voxel operations. Jones [Jon03] proposed a method to simulate the melting process taking into account the thermal flow and the latent heat caused by the phase change. These methods, however, neglect the liquids that are formed due to melting.

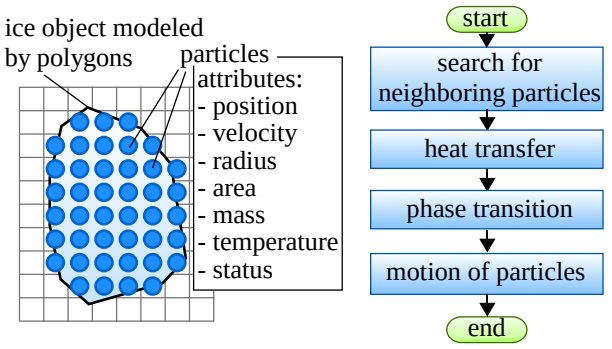
Carlson et al. [CMIT02] simulated the melting of solids such as waxes by treating solids as fluids with very high viscosities. Melting and flowing behaviors are simulated by solving the Navier-Stokes equations. Although this method can simulate the melting and flowing of high viscosity materials, it is not applicable to simulations of ice melting and of flows of meltwater, since the viscosity of water is low. Matsumura et al. [MT05] simulated the melting of ice, including the natural convection that occurs in the air surrounding the ice. Lossaso et al. [LIGF06] proposed a novel simulation method for the melting and burning of solid materials that takes into account the simulation of liquids and gases caused by the melting and burning processes. Fujisawa et al. [FM07] presented a method for simulating the melting of ice based on thermodynamics. Similar to the melting phenomena, Wojtan et al. [WCMT07] proposed a method for animating corrosion and erosion. Although these methods can simulate ice melting and flows of meltwater, they require several minutes for each time-step. Moreover, the formation of water droplets is not taken into consideration. To simulate meltwater droplets by extending these methods, the simulation grids would need to be finely subdivided at the cost of long computational time for the simulation.

Zhao et al. [ZWQ*06] proposed a lattice Boltzmann-based method to simulate melting and flowing, and accelerated the method using GPUs. Although this method can simulate the melting of solids at interactive frame rates, the flows of the droplets are not simulated. By increasing the number of lattice points, this method could simulate the flows of droplets, but again, this would increase the computational time.

Several methods have been proposed to simulate melting phenomena based on a particle-based approach. Terzopoulos et al. [TPF89] proposed a method to simulate the melting of deformable solids into fluids. Tonnesen [Ton91] modeled the melting phenomena by varying the dissociation energy of the Lennard-Jones functions. Muller et al. [MKN*04] presented a point-based animation method for the melting of elastic objects. Keiser et al. [KAG*05] proposed a unified Lagrangian approach to simulate melting and freezing. Paiva et al. [PPLT06] presented a non-Newtonian fluid animation for melting objects. Solenthaler et al. [SSP07] developed a unified particle model to simulate a wide variety of solid-fluid interactions, including melting, solidification, merging and splitting. Although these methods can simulate melting phenomena efficiently, the flows of meltwater droplets are not taken into consideration.

Only a few methods have been proposed to simulate freezing. Kim et al. [KAL06] proposed a physically-based simulation method to model ice formations by solving a Stefan problem. Madrazo et al. [MO08] proposed a method of simulating freezing that takes into account air bubbles in the ice.

Simulation of Water Droplets : Several researchers have developed methods for simulating the actions of droplets on surfaces. Dorsey et al. [DPH96] simulated the flows of water droplets for weathering simulations. Kaneda et al. [KKY93, KZYN96, KIY99] proposed efficient animation methods for water droplets moving on surfaces. Fournier et al. [FHP98] modeled the liquid in droplets based on the mass-spring model to simulate flows on surfaces. Yu et al. [YJC99] proposed a modeling method for static water droplets that takes into account the force of gravity. Tong et al. [TKY02] proposed a method for modeling water flows using metaballs that can preserve the original water volume. In recent years, Wang et al. [WMT05] have proposed a physically-based simulation method for water droplets by modeling the surface tensions between liquids and solids. Although this method can simulate flows on surfaces very realistically, the computational cost is quite expensive and interactive simulations are quite difficult.



(a) generation of particles. (b) processes for single time step.

Figure 1: *Overview of our method.*

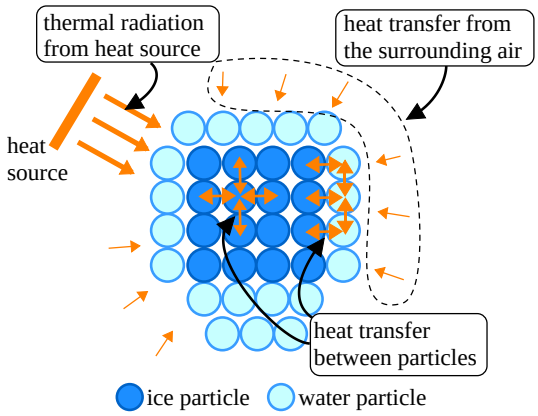


Figure 2: *Simulation of heat transfers.*

Muller et al. [MSKG05] proposed a simulation method for modeling surface tension, that is capable of modeling droplets. This method, however, requires the calculation of curvature, which is error-prone when the number of nearby particles is small. To address this problem, Becker et al. [BT07] improved the model for surface tension, similar to our method. However, their method is only applied to free surface flows and not to the interfacial tension between ice and meltwater. Clavet et al. [CBP05] proposed a double density relaxation scheme for viscoelastic fluids to simulate the formation of droplets. Although this method can model droplets on surfaces, it is not applicable for simulating flows of meltwater, since the viscosity of water is low.

3. Overview of Our Particle-based Melting Simulation

Fig. 1 shows an overview of our simulation method. An input to our system is an ice object modeled by a set of polygons. In a preprocessing step, the polygonal model is voxelized, as shown in Fig. 1(a), and a particle is generated at the center of each voxel. Although our voxel representation gives convincing results, our method can also represent the ice object with denser structures for the particles (e.g. a hexagonal close-packed structure). However, structures such as this generate many more ice particles, resulting in an increase in the simulation time, since these particles are used for the simulation of the ice melting. For each particle, a position, a velocity, an effective radius r_e, a representative surface area A, a mass m, temperature T, heat value, and the status of the particle (i.e. ice, water or rigid particles that represent rigid objects such as floors) are assigned. We set the effective radius r_e to the product of a user-specified constant and the voxel width d. The representative surface area A is set to the sum of the areas of the six faces of each voxel. The temperature T is calculated by using our simulation method for the melting process.

The melting process is simulated by computing the status (ice or water) and the motion of each particle. Fig. 1(b) shows the procedure for a single time-step of our simulation method. These processes are fully executed on the GPU (see Section 6). At each time-step, a set of neighboring particles are detected first. Then, heat transfer processes between the particles are calculated. The heat from the surrounding air and any other heat sources (such as an electric heater) are also taken into account in this process. Next, any phase transitions are simulated; that is, ice particles become water particles depending on their temperatures. The motion of the meltwater is then calculated based on the Smoothed Particle Hydrodynamics (SPH) method. The meltwater often moves along the surface of the ice object. This is simulated by taking into account the interfacial forces between the particles. The interfacial force works not only between water and ice particles, but also between water particles. Thus, several nearby water particles form a water droplet, and, when the force of gravity working on the droplet exceeds the interfacial forces, the droplet leaves the surface of the ice object and falls down.

4. Simulation of Heat Transfers

This section describes our method for simulating heat transfer processes. In this paper, we focus on the melting simulation due to heat transfer processes, and we leave the simulation of melting due to high pressure (e.g. regelation) for future work. As shown in Fig. 2, the temperatures of the particles are computed by taking into account three heat transfer processes: 1) heat transfers between particles (ice, water, and rigid particles), 2) heat transfers from the surrounding air, and 3) thermal radiation from external heat sources to particles.

The heat transfer between particles is calculated from:

$$\frac{\partial T_i}{\partial t} = c_d \sum_{j \in N_i} m_j \frac{(T_j - T_i)}{\rho_j} \nabla^2 W(r_{ij}, r_e), \qquad (1)$$

where T_i is the temperature of the particle i, t is the time, c_d is the thermal diffusion constant, (which depends on the status of particle i, i.e. ice, water or rigid), r_{ij} is the distance between the particle i and a nearby particle j, N_i is a set of particles whose distances from particle i are smaller than the effective radius r_e, and W is a smoothing kernel function which is the same as that used for computing viscosity, as proposed in [MCG03].

Although the heat transfer between ice and air can also be simulated by using Eq. (1), this requires many air particles around the ice particles, resulting in an increase in the simulation time. Instead, in order to achieve an efficient simulation, we approximate the temperature of the surrounding air as a constant ambient temperature T_{air}. Then, the heat transfer between the ice and the air is calculated from the Newton's law of cooling as $Q_i = h(T_{air} - T_i)\delta A$, where Q_i is the heat value received by an ice particle i, and h is the thermal conductivity. The value of δA represents the area of the ice surface in contact with the air at the position of particle i, and is calculated in the following way. As we mentioned in Section 3, the ice particles are generated at the centers of the voxels of the voxelized ice object. The surface area A assigned to each particle is the sum of the areas of the six faces of the voxel. Therefore, the number of faces touching the air among the six faces is calculated by subtracting the number of neighboring particles from 6. This indicates the area of the ice surface touching the air, δA, is calculated from $\frac{6-n_i}{6}A$, where n_i is the number of neighboring ice and water particles. If n_i exceeds 6, δA is clamped to 0. The increase in the temperature ΔT due to the heat value Q_i is calculated from $\Delta T = \frac{Q_i}{Cm}$, where C is the specific heat capacity of the ice, and m is the mass of the ice particle.

Our method assumes that the radiation energy from the external heat source is transferred by a set of particles similar to the process described by [FM07]. We call these particles *thermal photons*. The thermal energy from a heat source per unit time is carried by thermal photons containing a constant thermal energy E_p, calculated from $E_p = \frac{\varepsilon\sigma T_s^4 H dt}{N}$, where ε is an emissivity factor, σ is the Stefan-Boltzmann constant, T_s is the temperature of the heat source, H is the area of the heat source, dt is the time interval of the photon radiation, and N is the number of photons emitted at each time-step. Thermal photons are emitted from the heat source, and the hit tests between the photons and the particles are performed. Then, the number of thermal photons that hit each particle is calculated. The increase in the heat value δQ_i due to the thermal energy for a particle i is the product of the number of thermal photons that hit particle i and the thermal energy of each photon E_p. Our method computes all of these processes on the GPU (see Section 6).

After the calculation of the heat transfer, the phase transition of each particle is determined. If the temperature of the ice particle exceeds the melting point (for an ice particle, the melting point is 273 [K]) and the heat value exceeds the latent heat of fusion, then the particle changes its status and becomes a water particle. On the contrary, if the temperature of the water particle falls below the freezing point and the latent heat of solidification is released from the water particle, the water particle becomes an ice particle.

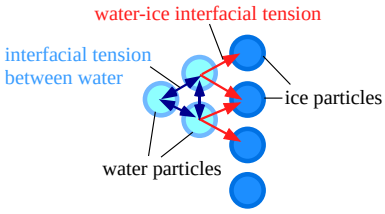


Figure 3: *Interfacial tension.*

5. Interactions between Ice and Meltwater

The motions of the water particles after the phase transition are calculated by using the Navier-Stokes equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f}, \quad (2)$$

$$\nabla\cdot\mathbf{u} = 0, \quad (3)$$

where $\mathbf{u}$ is the velocity, ρ is the density, p is the pressure, μ is the dynamic viscosity coefficient, and $\mathbf{f}$ represents the external forces. The Smoothed Particle Hydrodynamics (SPH) technique solves the Navier-Stokes equations by representing fluids as a set of particles. For particle-based simulations, the continuity equation shown here as Eq. (3) can be preserved automatically if no particles are deleted or inserted and if the particle masses are constant. The first and second terms on the right-hand side of Eq. (2) are referred to as the pressure force $\mathbf{f}_{press}$ and the viscosity force $\mathbf{f}_{vis}$, respectively. $\mathbf{f}_{press}$ and $\mathbf{f}_{vis}$ are computed by using the SPH method [MCG03]. The meltwater forms water drops, and these water drops often move along the ice surface before falling down. To simulate this behavior, our method takes into account both the interfacial tension between the water particles and that between water and ice particles (see Fig. 3). The interfacial tension $\mathbf{f}_i$ of the water particle i is calculated from the following equation:

$$\mathbf{f}_i = \sum_{j\in N_i^{water}} k_w \frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_j - \mathbf{x}_i||^2} + \sum_{j\in N_i^{ice}} k_{ice} \frac{\mathbf{x}_j - \mathbf{x}_i}{||\mathbf{x}_j - \mathbf{x}_i||^2}, \quad (4)$$

where $\mathbf{x}$ is the position of the particle, k_w is the coefficient of the interfacial tension between the water particles, k_{ice} is that for water-ice interfacial tension, and N_i^{water} (N_i^{ice}) is a set of water(ice) particles nearby to particle i. The interfacial force is incorporated into the external force $\mathbf{f}$ in Eq. (2).

Next, the motion of the entire ice object can be calculated in the following way when the ice object is not fixed. The ice object moves according to the forces working on the ice particles. The force on each ice particle i is calculated by

summing the force $\mathbf{f}_i^{fluid}$ from the fluids and the force $\mathbf{f}_i^{rigid}$ from other rigid bodies. The force from the fluids $\mathbf{f}_i^{fluid}$ is calculated by considering an ice particle i as if it were a water particle. The force $\mathbf{f}_i^{fluid}$ is computed by summing the pressure force $\mathbf{f}_{press}$ and the viscosity term $\mathbf{f}_{vis}$. The force $\mathbf{f}_i^{rigid}$ is calculated when the ice object collides with other objects such as floors and walls. $\mathbf{f}_i^{rigid}$ is the sum of the repulsive force, the frictional force, and the damping force. These are calculated by using the method proposed in [TSK07]. Then, the force $\mathbf{F}_{ice}$ on the center of mass of the ice object is calculated by summing these forces over the ice particles $\mathbf{F}_{ice} = \sum_{i \in N_{ice}} (\mathbf{f}_i^{fluid} + \mathbf{f}_i^{rigid})$, where N_{ice} is a set of ice particles belonging to the ice object. To handle rotation of the ice object, the torque τ_i due to the force on each ice particle i is computed from $\tau_i = (\mathbf{x}_i - \mathbf{x}_{cm}) \times (\mathbf{f}_i^{fluid} + \mathbf{f}_i^{rigid})$, where $\mathbf{x}_{cm}$ is the center of mass of the ice object. The total torque $\mathbf{T}_{ice}$ is calculated by summing the torque τ_i over all of the ice particles. The linear velocity $\mathbf{v}_{ice}$ is updated by using the force $\mathbf{F}_{ice}$. The angular velocity ω_{ice} is calculated from $\omega_{ice} = \mathbf{I}^{-1}\mathbf{L}$, where $\mathbf{I}$ is the inertia tensor and $\mathbf{L}$ is the angular momentum, which is updated from $\mathbf{L} + \mathbf{T}_{ice}\Delta t$ at each time-step.

6. GPU Implementation

The proposed simulation method is fully implemented on the GPU. To fully exploit the performance of the GPU, all of the physical values such as positions, velocities and temperatures of the particles are stored in global memory. For the sake of the memory efficiency and fast memory access, the values of the particle attributes that are identical for all particles (e.g. A and m) are stored in constant memory. For particle-based simulations, the calculations of the physical values require a search for nearby particles, as shown in Eqs. (1) and (4). Our method uses a uniform grid to accelerate the search process. The calculations of the forces and the heat transfer between particles are performed in parallel by launching each CUDA thread for each particle.

To calculate the thermal radiation from a heat source, hit tests between ice particles and thermal photons are performed. To accelerate the hit tests, our method exploits the voxel representation of the ice objects. That is, since each ice particle corresponds to each voxel, the hit test between the thermal photon and the ice particle is replaced with that between the thermal photon and corresponding voxel. The hit test between each thermal photon and each voxel is efficiently calculated by traversing voxels using 3DDDA algorithm [AW87]. Our method assigns each CUDA thread to each thermal photon, performs hit test in parallel, and stores the index of the hit ice particle. Then the number of thermal photons for each ice particle is calculated and the increased heat value due to thermal photons is calculated by multiplying the number of hit thermal photons and the energy of each photon.

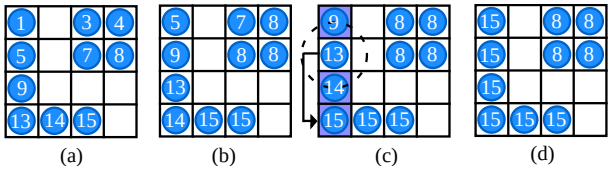


Figure 4: *Calculation of ice object each ice particle belongs to. (a) the number in the circle shows the initial index for each ice particle. (b)(c) index is updated by the maximum number of indices of nearby particles (within dotted circle) and that of the particle referred by the initial index . (d) numbers of indices are converged and two separate ice objects can be detected.*

The motions of ice objects also can be simulated by using the GPU. To simulate the motions of the ice objects, the force $\mathbf{F}_{ice}$ on the center of mass and the total torque $\mathbf{T}_{ice}$ for each ice object are calculated. To calculate these quantities, each particle must store the information concerning the ice object that the particle belongs to. The ice objects, however, can separate and merge due to melting and freezing. Therefore, our method has to update the information for the ice object that each particle belongs to at each time-step. To obtain this information, our method calculates the connected components of the ice object (see Fig. 4). First, each particle is assigned an index, as shown in Fig. 4(a). The index of each ice particle in the initial state is referred to as an initial index. Then, the index of each ice particle is updated iteratively by the maximum number of the indices of the nearby particles that belong to the same ice object in the previous time step (Fig. 4(b)). The indices of the ice particles that belong to the same ice object converge to the maximum number of the indices of those particles (Fig. 4(d)). Our method can detect the ice particles that belong to the same ice object by using the converged number. Although this method can detect the separation, it requires many iterations to converge. To reduce the number of the iterations, our method uses not only the indices of the nearby particles, but also the index of the particle referred by the initial index (Fig. 4(c)), since the particle of index i and the particle whose initial index is i belong to the same ice object. For example, the ice particle whose index is 13 in Fig. 4(c) is replaced with 15, since the numbers of the indices of the nearby particles are 9, 14 and that of the particle whose initial index is 13 is 15. Our method repeats these processes until the indices remain unchanged (Fig. 4(d)). These processes can be performed in parallel and are suitable for GPU computations.

The calculations of the center of mass $\mathbf{x}_{cm}$, $\mathbf{F}_{ice}$, $\mathbf{T}_{ice}$, and the inertia tensor $\mathbf{I}$ for each ice object require the sum of the physical quantities such as positions, forces, and torques over the ice particles. The summation of these physical quantities is computed by using a parallel reduction operation.

7. Rendering

To render the surfaces of the ice and fluids represented by the particles, a density distribution is assigned to each particle, and the surfaces of the ice and the fluids are represented by iso-surfaces extracted from the density distribution. Our method provides two rendering methods; one is an off-line rendering method and the other is an interactive rendering method for previewing the simulation results. In the off-line rendering method, the iso-surfaces of the ice and the water are extracted by using the marching cube technique. Then, the extracted iso-surfaces are rendered by using the ray tracing technique. In the interactive rendering method, in order to achieve interactive feedback, our method employs a GPU-based ray casting method similar to [KSN08].

8. Results and Discussion

Fig. 5 shows a simulation of a melting ice-dragon and the flows of the meltwater drops running away from it. The ice-dragon melts due to heat from the surrounding air. The average simulation time is 37ms on a standard PC (CPU: Core i7 Extreme, GPU: GeForceGTX470). Our method uses CUDA and Cg for GPU implementation. The images are rendered by using Pov-ray, except for Fig. 12. The parameters used in the simulation are shown in Table 1.

Fig. 6 shows melting icicles. The icicles are heated by the surrounding air. The meltwater flows down the icicles and forms droplets at the tips of the icicles. The average computational time of the simulation for the single time-step is 14msec.

Fig. 7 shows an animation in which water is poured into a glass containing four ice-cubes. The computational time for each time-step varies from 17msec to 84msec depending on the number of water particles.

Fig. 8 visualizes the temperatures of particles in Fig. 7. Fig. 8 is a cross section of the simulation space. As shown in Fig. 8, the ice particles touching the air, water, and glass are heated and the heat transfers to the inside ice particles.

Fig. 9 shows an animation of an ice-bunny melting due to a heat source. The number of photons emitted from the heat source at each time-step is 100k. The average computational time for the simulation is 45msec. The back of the ice-bunny has melted due to the thermal radiation emitted from the heat source.

Fig. 10 shows an animation of an ice-buddha melting due to hot water. The hot water thaws the legs of the ice-buddha. Then, the pedestal separates from the body, and falls down into the water. The pedestal and the body float on the water. The total number of particles is 60k. The average computational time for the simulation is 63msec.

Fig. 11 shows the formation of an ice spike by freezing water particles. The water droplets collide with the ice and the droplets are cooled due to the ice as they flow along the surface of the ice. The total number of ice particles is 22k, and the average simulation rate is 6msec.

Fig. 12 shows images of melting ice-dragon and melted water droplets that were simulated and rendered using our method. The number of particles in Fig. 12 is 60k, and the computational time, including simulation and rendering, is 8.0 fps. Since our method calculates the iso-surface per pixel, our method can capture surfaces of small water droplets interactively.

Fig. 13 shows a comparison between our interfacial tension model and Muller's surface tension model [MCG03]. The number of ice particles is 44k. In Muller's model, the meltwater runs along the surface of the upper half of the ice. However, the meltwater droplets fall at the corner of the ice, and do not run along the surface of the bottom half of the ice. On the other hand, our model can represent the flows of the meltwater along the surface of the ice, and also the formation of water droplets at the tip of the ice.

We compare the computational time of our GPU implementation with the previous melting simulations. Most relevant previous method is Solenthaler's particle-based melting simulation method [SSP07]. They reported that the simulation for 40k particles runs at about 500msec on an Intel iMac 2GHz. They also reported that an acceleration by a factor of 10 is feasible using a combination of fast incremental neighbor search, parallelism and hardware accelerated techniques. On the other hand, the simulation time for 60k ice-buddha melting in Fig. 10, including solid-fluid interaction and separation simulation is 63 msec. The computational time of our method is almost proportional to the number of particles. Therefore, the estimated simulation time for 40k particles is about 42 msec. Even if the Solenthaler's method is accelerated by a factor of 10 by optimization and GPU implementation, it seems that our method is faster than the Solenthaler's method. Please note that Solenthaler's method does not deal with the flows of the meltwater. In other relevant previous methods such as [LIGF06] and [FM07], the computational time for each time step is several minutes. Our method runs at less than 100 msec in the examples shown in this section. Therefore, even if these methods are accelerated by two orders of magnitude using GPUs, it seems that our method is faster than the previous melting simulation methods.

Although our method can create convincing melting ice animations, it still has several limitations. First of all, our method assumes that the change in the air temperature as it is cooled by the ice is small, and thus it does not calculate the flow of air. Therefore, our method does not handle the motion of the surrounding air due to heat convection. However, this phenomenon can be considered by combining our method with traditional fluid simulation methods such as [Sta99]. Secondly, our method considers only single-bounce photons for the thermal radiation calculations. However, our method can also calculate multiple-bounces of photons by repeating the GPU-based hit tests described in Section 6,

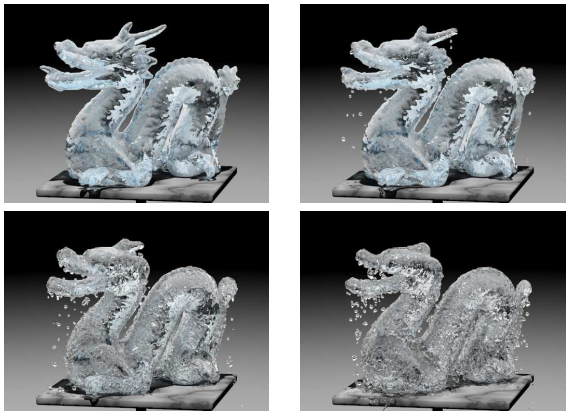


Figure 5: *Melting ice-dragon (240k particles).*

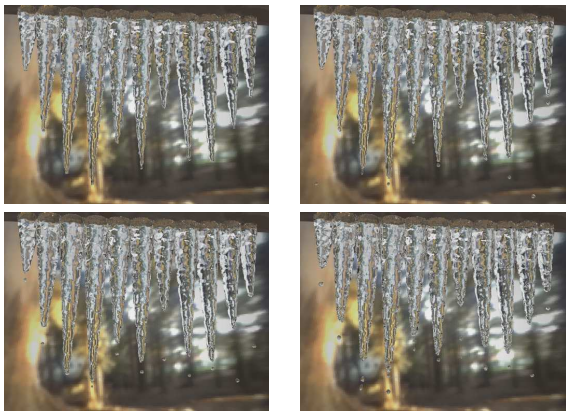


Figure 6: *Melting icicles (102k particles).*

though at the cost of increased computational time. This means that there is a trade-off between the computational time and the accuracy of the simulation. Thirdly, some grid artifacts can be seen in the interactive rendering. This is because the radius of the metaball is set to be small to achieve interactive feedback. The iso-surfaces can be smoother if the radius of the metaball is set to be large, which also increases the overlapping regions of the metaballs. This also increases the number of times of depth peeling, resulting in the lower performance. This is the trade-off between the image quality and the rendering time. Finally, our method does not handle the destruction of the ice due to melting. We believe that this can be simulated by incorporating a meshless fracturing simulation method [PKA*05] into our method.

9. Conclusions and Future Work

We have proposed a fast simulation method for the melting of ice and the interactions of ice and meltwater. We introduced a simple but efficient interfacial tension model between water-water and water-ice particles. Our interfacial tension model can simulate the flows on the ice and water

Figure 7: *Pouring water (200k particles) into a glass (65k particles) with four ice-cubes (32k particles).*

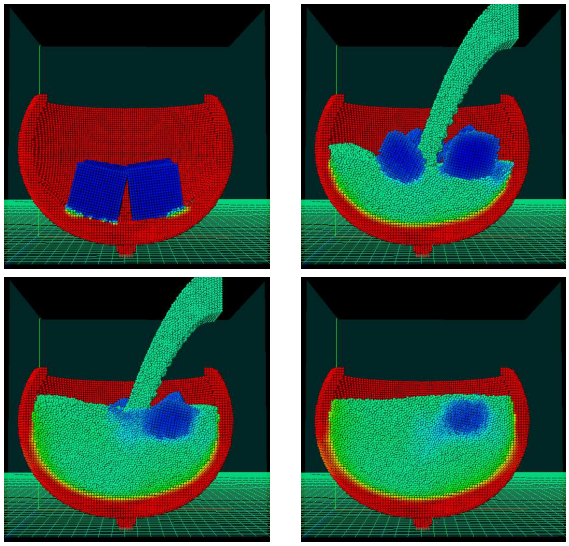


Figure 8: *Visualization of heat transfer of Fig. 7. The color of each particle indicates the temperature, where red indicates high temperature and blue shows low temperature.*

droplets, and it can easily be incorporated into the SPH. A wide variety of calculations, including heat transfer, interactions between ice and fluids, the separation of the ice due to melting, and rendering, are fully implemented on GPUs. Our method can simulate the melting of ice consists of 240k particles interactively. We believe that our interactive melting simulation is beneficial to interactive applications such as VR and games. Moreover, our method can be applied to the simulation of freezing.

In future work, we would like to simulate the natural convection of air by combining our method with the grid-based simulation of air.

Figure 9: *Melting ice-bunny (141k particles) due to the heat source over the back of the ice-bunny.*

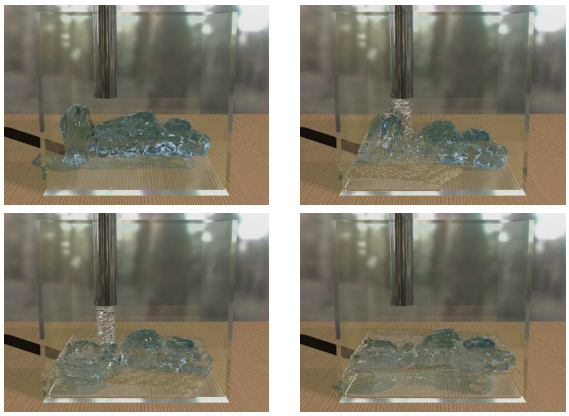


Figure 10: *Melting ice-buddha due to hot water. The legs of the buddha are melted due to the hot water, and then the body and the pedestal are separated.*

Acknowledgement

We would like to thank anonymous reviewers for the useful comments. This research was partially supported by Hayao Nakayama Foundation for Science & Technology and Culture.

Figure 11: *Formation of an ice spike by freezing water droplets.*

Figure 12: *Interactive rendering and simulation of melting ice-dragon due to an external heat source represented by a red square.*

(a) our model (b) Muller's model

Figure 13: *Comparison of melting simulations between our interfacial tension model and Muller's model.*

References

[AW87] AMANATIDES J., WOO A.: A fast voxel traversal algorithm for ray tracing. In *Eurographics 1987* (1987), pp. 3–10.

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007), pp. 209–217.

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 219–228.

[CMIT02] CARLSON M., MUCHA P., III B. V. H., TURK G.: Melting and flowing. In *Proc. ACM SIGGRAPH Symposium on Computer Animation* (2002), pp. 167–174.

[DPH96] DORSEY J., PEDERSEN H. K., HANRAHAN P.: Flow

Table 1: *Parameter setting of melting simulation.*

parameter	meaning	value
Δt	time-step	0.003
d	voxel width	1
m	particle mass	0.0008
r_e	effective radius	1.1
c_d	ice thermal diffusion constant	0.5
c_d	water thermal diffusion constant	0.1
h	thermal conductivity	0.00267
σ	Stefan Boltzmann constant	5.67×10^{-8}
k_w	interfacial tension coefficient for water-water particles	0.5
k_{ice}	interfacial tension coefficient for water-ice particles	2.0

and changes in appearance. In *Proc. SIGGRAPH 96* (1996), pp. 411–420.

[FA01] FUJISHIRO I., AOKI E.: Volume graphics modeling of ice thawing. In *Volume Graphics* (2001), pp. 69–80.

[FHP98] FOURNIER P., HABIBI A., POULIN P.: Simulating the flow of liquid droplets. In *Proc. Graphics Interface 98* (1998), pp. 133–142.

[FM07] FUJISAWA M., MIURA K. T.: Animation of ice melting phenomenon based on thermodynamics with thermal radiation. In *Proc. GRAPHITE 2007* (2007), pp. 249–256.

[Jon03] JONES M. W.: Melting objects. *The journal of WSCG 11*, 1 (2003), 247–254.

[KAG*05] KEISER R., ADAMS B., GASSER D., BAZZI P., DUTRE P., GROSS K.: A unified lagrangian approach to solid-fluid animation. In *Eurographics Symposium on Point-Based Graphics 2005* (2005), pp. 125–133.

[KAL06] KIM T., ADALSTEINSSON D., LIN M.: Modeling ice dynamics as a thin-film stefan problem. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 167–176.

[KIY99] KANEDA K., IKEDA S., YAMASHITA H.: Animation of water droplets moving down a surface. *The Journal of Visualization and Computer Animation 10*, 1 (1999), 15–26.

[KKY93] KANEDA K., KAGAWA T., YAMASHITA H.: Animation of water droplets on a glass plate. In *Proc. Computer Animation 93* (1993), pp. 177–189.

[KSN08] KANAMORI Y., SZEGO Z., NISHITA T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum (Proc. of Eurographics 2008) 27*, 3 (2008), 351–360.

[KZYN96] KANEDA K., ZUYAMA Y., YAMASHITA H., NISHITA T.: Animation of water droplet flow on curved surfaces. In *Proc. Pacific Graphics 96* (1996), pp. 50–65.

[LIGF06] LOSASSO F., IRVING G., GUENDELMAN F., FEDKIW R.: Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics 12*, 3 (2006), 343–352.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 141–151.

[MO08] MADRAZO C., OKADA M.: Physically based modeling of ice with bubbles. In *Eurographics Short Papers* (2008), pp. 13–16.

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 237–244.

[MT05] MATSUMURA M., TSURUNO R.: Visual simulation of melting ice considering the natural convection. In *SIGGRAPH 05: ACM SIGGRAPH 2005 Sketches* (2005), p. 61.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRE P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005), 957–964.

[PPLT06] PAIVA A., PETRONETTO F., LEWINER T., TAVARES G.: Particle-based non-newtonian fluid animation for melting objects. In *Sibgrapi 2006* (2006), pp. 78–85.

[SSP07] SOLENTHALER B., SCHLAFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007), 69–82.

[Sta99] STAM J.: Stable fluids. In *Proc. SIGGRAPH 99* (1999), pp. 121–128.

[TKY02] TONG R., KANEDA K., YAMASHITA H.: A volume-preserving approach for modeling and animating water flows generated by metaballs. *The Visual Computer 18*, 8 (2002), 469–480.

[Ton91] TONNESEN D.: Modeling liquids and solids using thermal particles. In *Proc. Graphics Interface 91* (1991), pp. 255–262.

[TPF89] TERZOPOULOS D., PLATT J., FLEISCHER K.: Heating and melting deformable models (from goop to glop). In *Proc. Graphics Interface 89* (1989), pp. 219–226.

[TSK07] TANAKA M., SAKAKI M., KOSHIZUKA S.: Particle-based rigid body simulation and coupling with fluid simulation. *Transactions of Japan Society for Computational Engineering and Science*, Paper No.20070007 (2007).

[WCMT07] WOJTAN C., CARLSON M., MUCHA P. J., TURK G.: Animating corrosion and erosion. In *Eurographics Workshop on Natural Phenomena* (2007), pp. 15–22.

[WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. *ACM Trans. Graph. 24*, 3 (2005), 921–929.

[YJC99] YU Y.-J., JUNG H.-Y., CHO H.-G.: A new water droplet model using metaball in the gravitational field. *Computer and Graphics 23*, 2 (1999), 213–222.

[ZWQ*06] ZHAO Y., WANG L., QIU F., KAUFMAN A., MUELLER K.: Melting and flowing in multiphase environment. *Compute and Graphics 30*, 4 (2006), 519–528.