Vis Comput (2014) 30:531–538
DOI 10.1007/s00371-013-0878-1

ORIGINAL ARTICLE

Ice melting simulation with water flow handling

Shing-Yeu Lii · Sai-Keung Wong

Published online: 14 September 2013

Abstract In this paper, we propose a new approach based on a particle-based model for ice melting simulation. Each particle has an attribute called virtual water. The amount of the virtual water of an ice particle indicates the amount of water surrounding the ice particle. The transfer of the virtual water is performed between the exterior ice particles so as to simulate the thin layer of water flow on the surface. Our approach also handles the transition between the virtual water and the water particles. We compute the isosurface of a density field defined by the ice particles and the virtual water. A simple ray tracing method is adopted for rendering the objects. We report the experimental results of several ice melting simulations with water flow and water drops.

Keywords Ice melting · Marching cubes · Metaballs

1 Introduction

Ice absorbs heat from the surroundings, such as air and radiation. The heat of ice exchanges internally between molecules. When ice absorbs enough energy, the molecular bonds of the molecules are broken and ice melts into water. While ice melts gradually into water, a thin layer of water is formed and flows over the surface of ice. Water drops are formed at the lower parts of ice when the water accumulates. In this paper, We propose a new approach based on a particle-based model for simulating ice melting. Our approach handles the water flowing and the formation of water drops in a physically plausible manner.

In particle-based approaches, such as [7], an ice object and the meltwater are represented as a set of ice particles and a set of water particles, respectively. The phase transition between ice particles and water particles is achieved by converting ice particles into water particles and vice versa. The thin layer of water on the surface of the ice object is simulated as a set of densely sampled water particles. This increases the computation cost. Furthermore, noticeable artifact is observed if ice particles are converted immediately into water particles. This is because the part of the isosurface around the ice particles changes immediately to its new position. In our approach, each ice particle has an attribute called *virtual water*. The virtual water of an ice particle indicates the amount of water around the ice particle. The virtual water can be transferred between the ice particles. In this way, we can model the flowing of the thin layer of water on the surface. Figure 1 shows our simulated result of a bunny model. The bunny model shrinks inwards smoothly while it melts into water. Figure 6 shows the comparison of our method with [7].

The major contributions of our approach are stated as follows:

1. We incorporate an attribute called virtual water for ice particles. The transfer of the virtual water between ice particles is performed for simulating the water flow on the surface. The density field is defined by the ice particles and the virtual water.
2. We develop a scheme for exchanging between virtual water and water particles.

Electronic supplementary material The online version of this article (doi:10.1007/s00371-013-0878-1) contains supplementary material, which is available to authorized users.

S.-Y. Lii · S.-K. Wong (✉)
Department of Computer Science, National Chiao Tung University, Hsinchu, R.O.C. Taiwan
e-mail: cswingo@cs.nctu.edu.tw

S.-Y. Lii
e-mail: f030258631@hotmail.com

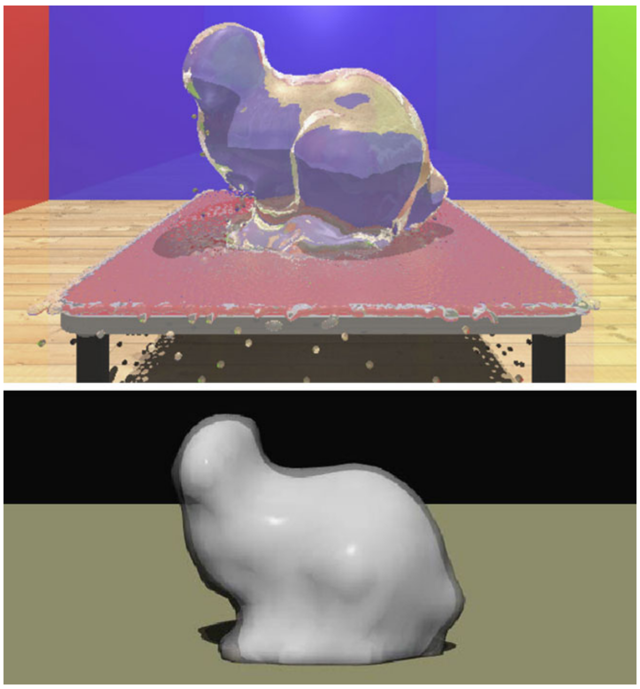

Fig. 1 (*Top*) A melting bunny model on the table. The light source is placed at the back of the bunny model. The water drops and the thin layer of water on the surface are simulated. (*Bottom*) A layer of virtual water surrounding the surface of the model

2 Related work

The basic concept for simulating object melting is to utilize the thermodynamic heat transfer to conduct heat and to melt objects. Terzopoulos et al. [15] modeled a melting object as a mass-spring system in which the stiffnesses of the springs are inversely related to the temperature of the elements. Zhao et al. [17] simulated objects with different materials that melt and flow in multiple phases. The method is developed based on a modified lattice Boltzmann method.

Fujishiro et al. [4] simulated the melting of ice by using the morphology. The method voxelizes the polygonal model and performs heat transfer between voxels. However, the method does not handle the fluid that is formed due to melting. Carlson et al. [12] simulated melted solid objects such as wax or clay, and handled melted fluid by using the Navier–Stokes equations. The method can simulate melted fluid for the soil-like objects with high stickiness.

Ice melting techniques are proposed in [6, 10, 16]. These methods also handle the motion of water, but they take several minutes to compute for each frame. Fujisawa et al. [6] adopted a step function called the Volume-of-Fluid with advection for tracking the free surface of an ice object. To simulate a flow thinner than the grid width, a grid cell is subdivided into smaller grid cells. Their method does not handle water drops and mass conservation is not guaranteed.

Solenthaler et al. [14] developed a unified particle model to simulate solid-fluid interactions, including melting, solidification, merging and splitting. In the approach proposed by Arash et al. [1], the fluid model is simulated based on an Eulerian grid model [5] and the solid is modeled as a set of nodal masses connected by springs.

Iwasaki et al. [7] proposed a particle-based method which is implemented on GPUs. The method handles the interaction between ice, water, and water droplets. We adopt their particle-based model in our approach. However, our treatment for handling water flow is different from theirs. Their method converts ice particles into water particles directly if certain conditions are satisfied. It requires a large number of water particles to simulate the thin layer of water on the ice surface. In contrast, our approach converts ice particles into virtual water gradually. The isosurface of the ice model is computed based on the ice particles and virtual water.

3 Definitions and overview

We extend the particle-based model [7] in which ice particles and water particles represent ice and meltwater, respectively. The attributes of a particle a include its position $\mathbf{x}_a$, mass m_a, latent heat and temperature. In our model, an ice particle has another attribute called virtual water. The amount of virtual water of an ice particle indicates the amount of water surrounding the ice particle. The volume of the virtual water represents the thin layer of water surrounding the ice particles. The masses of the ice particles and the virtual water can be exchanged according to their temperature differences and heat energy. Our motivation for adopting virtual water is that we do not need to use densely sampled water particles to simulate the water flow on the ice surface. Figure 2 gives an illustration. The water particles are seen individually on the ice surface if water particles are not densely sampled.

We voxelize an input model by applying the method proposed in [13]. Figure 3 shows the voxel structure on the left hand side. An ice particle is created at the center point of each voxel. We can refer to an ice particle a based on its coordinates (i, j, k) or vice versa. The default volume of an ice particle is set to V_{init}. An ice particle is *interior* if there are ice particles at all its six sides; otherwise, it is *external*. The virtual water is transferred between the external ice particles. A photon mapping method is adopted for thermal radiation from the source to the particles. An ice particle is gradually converted into virtual water when some conditions are satisfied. If the volume of the virtual water of an ice particle is larger than a predefined threshold, a fraction of the virtual water is converted into water particles.

There are five stages in the process of ice melting: heat transfer, phase transition, virtual water transfer, computation of the motion of particles, and rendering. The density field is computed based on the ice particles and the virtual water.

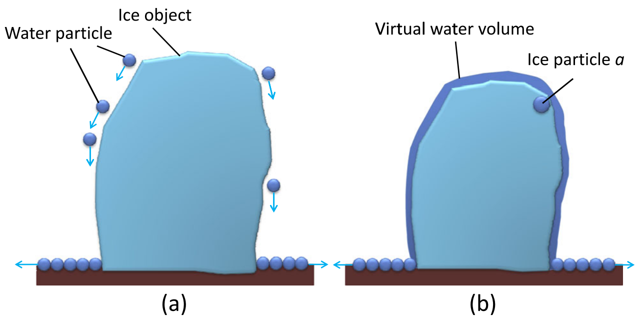

Fig. 2 With and without using virtual water. (**a**) Without using virtual water. The water particles are seen individually on the ice surface. (**b**) Our approach with using virtual water. A layer of virtual water volume surrounds the ice object. *The arrows* indicate the moving directions of the water particles

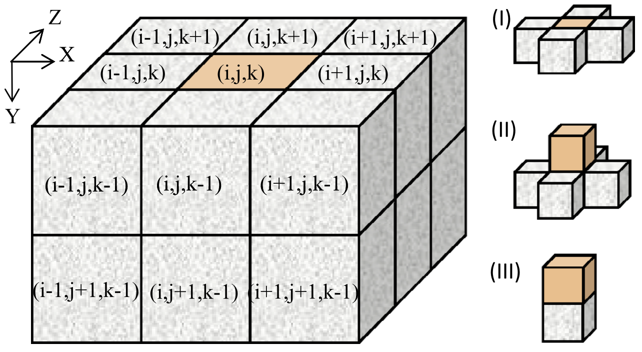

Fig. 3 (*Left*) Organization of voxels; (*right*) three transfer types for the transfer of the virtual water. *Type I*: the virtual water moves to the external ice particles at the same level; *Type II*: the virtual water moves to the diagonal external ice particles at the next level; *Type III*: the virtual water moves to the external ice particle at the next level

Then the isosurface of the density field is constructed by using the marching cubes algorithm. After that we use a ray tracer to render the isosurface and water particles. The water particles are rendered as metaballs [8].

4 Heat transfer and phase transition

The heat transfer from the heat source to the particles is performed based on the photon mapping method [6]. The energy of a photon is given by

$$E_p = \frac{\epsilon \sigma T^4 A \Delta t}{N}, \tag{1}$$

where T is the temperature, Δt is the time interval of the photon radiation, N is the number of photons emitted from the heat source of area A per unit time, $\sigma = 5.67 \times 10^{-8}$ [W/m^2 K^4] is the Stefan–Boltzmann constant, and ϵ is the emissivity of the heat source. Some energy, called latent heat of fusion, is consumed to break down the molecular bonds in the solid when the phase change occurs from solid (ice) to liquid (water).

It is not appropriate for converting the ice particle immediately into water particles until the ice particle absorbs enough latent heat of fusion. In our approach, an ice particle a represents a large number of molecules. When the ice particle absorbs latent heat of fusion, the bonds of some molecules of the ice particle are broken and their states change to liquid. This usually occurs at the surface of the ice model. Hence, we gradually convert a fraction of an ice particle into virtual water based on the ratio of the current heat of fusion to the maximum heat of fusion of the ice particle.

The ice particle melts fully into water if its temperature is 273 [K] and its heat energy is equal to the latent heat of fusion. The latent heat of fusion required to melt the entire ice particle is equal to $E_{a,\text{total}} = m_a E_w$, where E_w is the specific heat energy of ice. The ice particle a absorbs heat value ΔQ_a at a time interval. The heat value ΔQ_a is calculated as $N_a E_p$, where N_a is the number of thermal photons that hit ice particle a. If the temperature of the ice particle is below 273 [K], the increase ΔT of temperature due to the heat value ΔQ_a is computed as $\Delta T = \frac{\Delta Q_a}{m_a C}$, where C is the heat capacity of ice. The temperature of ice particle a increases up to 273 [K]. After that the absorbed thermal energy ΔE_a increases the latent heat of ice particle a. The ratio $\Delta E_a^{\text{ratio}}$ is computed as

$$\Delta E_a^{\text{ratio}} = \frac{\Delta E_a}{E_{a,\text{total}}} \tag{2}$$

which indicates the percentage of the ice particles that are converted into virtual water at the current time step. Then the following formula is adopted for converting the fraction of an ice particle into the corresponding volume of the virtual water:

$$\Delta V_a = \gamma V_a \Delta E_a^{\text{ratio}} \tag{3}$$

where ΔV_a is the change of the volume of the virtual water in the current time step, γ is the ratio of the water to ice volume, and V_a is the volume of the ice particle. We add ΔV_a to the virtual water of ice particle a and subtract $\frac{\Delta V_a}{\gamma}$ from V_a. The value of γ is around 0.92 under the normal pressure condition.

4.1 Density field

We adopt a distance-like function which is similar to the one proposed in [2] for computing the isosurface of fluid. The density field function $\Phi(\mathbf{x})$ is defined as

$$\Phi(\mathbf{x}) = \sqrt{\sum_a \left(\sqrt[3]{\frac{X_a}{V_{\text{init}}}} \left(1 - \frac{|\mathbf{x} - \mathbf{x}_a|}{h} \right) \right)^2} \tag{4}$$

where $X_a = V_a + Z_a$, h is the particle interactive radius and Z_a is the volume of the virtual water of ice particle a. For smaller X_a, less contribution is made by ice particle a to the density field. The threshold θ for computing the isosurface can be selected around 0.5. To see the rationale for defining $\Phi(\mathbf{x})$, we consider a single particle a centered at the origin. In this case, the isosurface is a sphere. Let R_1 and R_2 be the radii of the spheres with and without virtual water, respectively. Assume that $\theta = 0.5$, $V_a = V_{\text{init}}$, and Z_a is relatively much smaller than V_{init}. Then $R_1 = (1 - (\frac{X_a}{V_{\text{init}}})^{-\frac{1}{3}}\theta)h$ and $R_2 = (1 - \theta)h$. The ratio of the volumes of the two spheres with and without virtual water is equal to $\frac{\frac{4}{3}R_1^3\pi}{\frac{4}{3}R_2^3\pi} \approx \frac{V_{\text{init}}+Z_a}{V_{\text{init}}}$. The result shows that the proposed density function is appropriate for computing the density field. Please see the appendix for deriving the density function.

5 Virtual water transfer and motion computation

An interior ice particle does not contain any virtual water. Hence, the transfer of virtual water occurs for the external ice particles. We rely on two assumptions to perform the transfer of the virtual water. The first assumption is that due to the gravitational force, the virtual water moves to lower regions. The second assumption is that the virtual water of a particle moves to another particle at the same level (i.e. with the same y-coordinate) if the former has more amount of virtual water than that of the latter.

We build an update list U_a for each particle $a(i, j, k)$ and then update the amount of virtual water of the particles based on the update list. An element of an update list has three components: source particle, destination particle, and transfer type. There are three transfer types to consider for external ice particle $a(i, j, k)$, as illustrated in Fig. 3.

1. Type I: If $a(i, j, k)$ has neighboring ice particles at the same level and they have less amount of virtual water than that of $a(i, j, k)$, they are added to U_a.
2. Type II: The virtual water can move from $a(i, j, k)$ to the diagonal particles at the lower level if there are no other particles between $a(i, j, k)$ and the diagonal particles. For example, if there is no particle at $(i, j, k-1)$, then the particle at $(i, j+1, k-1)$ is added to U_a.
3. Type III: The virtual water of $a(i, j, k)$ moves downwards to the particle at $(i, j+1, k)$ if there are no particles at $(i-1, j, k)$, $(i, j, k-1)$, $(i+1, j, k)$ or $(i, j, k+1)$. In this case, the particle at $(i, j+1, k)$ is added to U_a.

After we have built the update list for each particle, we then update the amount of the virtual water of the ice particles. Denote Z_b as the volume of the virtual water of ice particle b. Consider the update list U_a of ice particle a. We process each element $(a, b, \text{type}(a, b))$ of U_a. For Type I, we update Z_a to $Z_a - \lambda_{\text{type}(a,b)}(Z_a - Z_b)\Delta t$ and Z_b to $Z_b + \lambda_{\text{type}(a,b)}(Z_a - Z_b)\Delta t$. For Type II and III, we update Z_a to $Z_a - \lambda_{\text{type}(a,b)} Z_a \Delta t$ and Z_b to $Z_b + \lambda_{\text{type}(a,b)} Z_b \Delta t$. The value of $\lambda_{\text{type}(a,b)}$ lies within $(0, 1)$.

5.1 Water particles

After the transfer of the virtual water is performed, we then proceed to the step for creating water particles. The water particles have the same volume and we denote V_{wp} as the volume of a water particle. By setting V_{wp} to a smaller value, more water particles will be created. This is a tradeoff between speed and accuracy.

Water particles are created if the volume of the virtual water of an ice particle is larger than V_{wp}. This usually occurs for the external ice particles at the lower parts (e.g., drooping tip of icicles) of the ice model. The virtual water of such external ice particles is not able to be transferred to other ice particles. Of course, if the ice object melts fast, water drops also appear at the other places.

After a water particle is created, the amount of the virtual water of the external ice particle is reduced by V_{wp} and a water particle is created at the center of the external ice particle. If the water particles were created at the sides of the ice particles, the water particles would suddenly appear at the surface of the ice model. This would lead to unnatural appearance of the simulation. Hence, our method sets the position of the new water particle as the center of the ice particle. An initial velocity is assigned to the water particle and its direction points to the exterior of the ice model. We also adopt the method presented in [7] for handling the interfacial interaction between the water particles and the ice particles. Spring forces are added between ice particles and water particles. Please refer to [7] for details.

5.2 Handling special cases

There are two special cases that we need to handle. In the first case, when an ice particle absorbs enough amount of latent heat and it is fully converted into virtual water; and then the ice particle is removed. However, the virtual water associated with the ice particle would be lost. Therefore, we transfer the virtual water of the ice particle to the neighboring ice particles before that ice particle is removed.

In the second case, some water particles drop onto ice particles and stay on the ice particles. We would see clearly tiny spheres on the surface since the water particles are rendered as metaballs. To solve this problem, we convert the water particles into the virtual water and allocate evenly the virtual water to the nearby external ice particles.

6 Experiments and results

The experiments were performed on an Intel® Core™ i7-2600 CPU and 3.40 GHz with 16.0 GB RAM, and NVIDIA GeForce GTX 670. In our system, all the steps were implemented on GPUs with OpenCL 1.1 except for the transfer of virtual water and the marching cubes algorithm. These two steps were implemented on CPU.

We adopted ray tracing to render the isosurface and water particles. The water particles were rendered as metaballs. Hence, we did not need to increase the grid resolution for capturing the shape of water drops. We built the linear bounding volume hierarchy (LBVH) for the isosurface [9] so as to accelerate the process of ray tracing. We adopted the method proposed in [8] for ray tracing the metaballs. This method efficiently computes the intersection between rays and the isosurface of the metaballs.

There are four benchmarks: Cube, Sphere, Bunny, and Dragon. Each benchmark has 1000 frames. Table 1 shows the information about the model complexities, the initial number of the ice particles after voxelization, and the grid resolution. For the simple models Cube and Sphere, the grid resolution is $32 \times 32 \times 32$. For the complex models Bunny and Dragon, the grid resolution is $64 \times 64 \times 64$. Figure 4 shows the snapshots of the four benchmarks. The ice models shrink inwards smoothly while they melt into water. The water accumulates at the bottom of the models.

Table 2 shows the simulation timings of the benchmarks, the average number of generated water particles, and the average number of triangles (constructed by the marching cubes algorithm). The two major time-consuming steps are the computation of particle motion (up to 263.81 ms in Dragon) and the marching cubes construction (up to 268.86 ms in Dragon). The cost of computing heat transfer depends on the number of particles. For example, it takes 144.58 ms on average to compute heat transfer in Dragon. The average timings of performing phase transition range

Table 1 Model complexities

Benchmarks	# Triangles	# Vertices	# Ice particles	# Grid resolution
Cube	8	12	13,248	$32 \times 32 \times 32$
Sphere	960	482	7,399	$32 \times 32 \times 32$
Bunny	69,668	34,837	22,795	$64 \times 64 \times 64$
Dragon	200,000	50,000	53,351	$64 \times 64 \times 64$

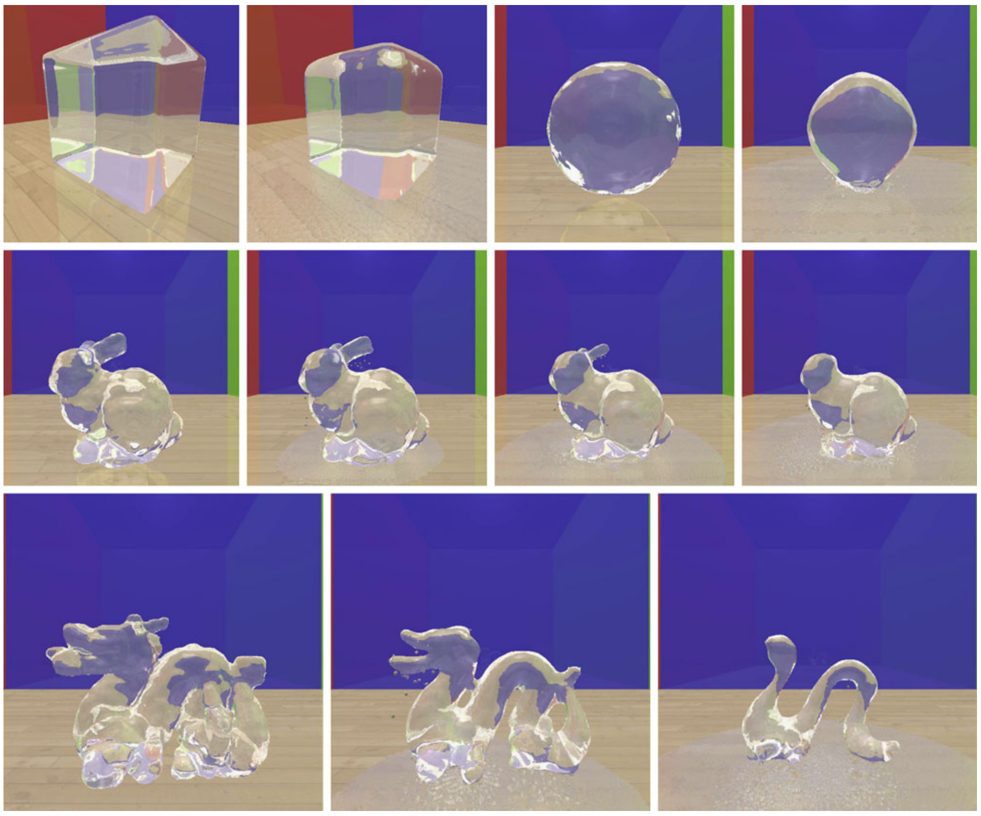
Fig. 4 A series of snapshots of the benchmarks. (*From top to bottom*) Cube, Sphere, Bunny, and Dragon. There are highlights on the surfaces of the metaballs

Table 2 The experimental results of ice melting simulation. First two columns: the average number of water particles and number of triangles constructed by using the marching cubes method; Remaining columns: the average timings for different steps

	# Water particles	# MC tri.	Heat transfer (ms)	Phase transition (ms)	Transfer of virtual water (ms)	Motion of particles (ms)	MC construction (ms)	Total time (ms)
Cube	3348.5	9190.6	12.24	1.01	0.57	23.16	29.54	66.52
Sphere	3795.6	6770.7	6.43	0.98	0.38	15.99	21.94	45.72
Bunny	6211.6	25777.0	33.25	1.40	1.15	54.62	124.54	214.96
Dragon	17390.1	35161	144.58	3.87	4.14	263.81	268.86	685.26

Table 3 Timings of different stages in rendering

	BVH construction for mesh (ms)	BVH construction for metaballs (ms)	Ray tracing (ms)	Total time (ms)
Cube	87.72	75.83	258.52	422.07
Sphere	95.75	70.43	593.44	759.62
Bunny	84.60	53.62	642.01	780.23
Dragon	109.59	79.33	1086.67	1275.59

from 0.98 ms to 3.87 ms. For the transfer of virtual water, the average timings range from 0.38 ms to 4.14 ms.

Table 3 shows the timings for construction of BVHs and ray tracing. Here the image resolution is 720 × 720. Our ray tracer uses reflection ray, refraction ray, and shadow ray. There are three parts for the entire process, BVH construction for the triangular mesh, BVH construction for metaballs, and ray tracing. It takes longer time for higher number of water particles and number of triangles.

Our method mixes with metaballs and triangular meshes for rendering the simulated results. The advantage of our approach is that we do not need to use finer grids to capture the shape of smaller water drops. In Fig. 5, we compare the simulated results with and without using metaballs at the same grid resolution. The left side of Fig. 5 shows the result without using metaballs. Some water drops do not appear properly as the grid resolution is not small enough. On the other hand, water particles are rendered as metaballs in the right side of Fig. 5. We can clearly see the water droplets falling down and their shapes are smooth.

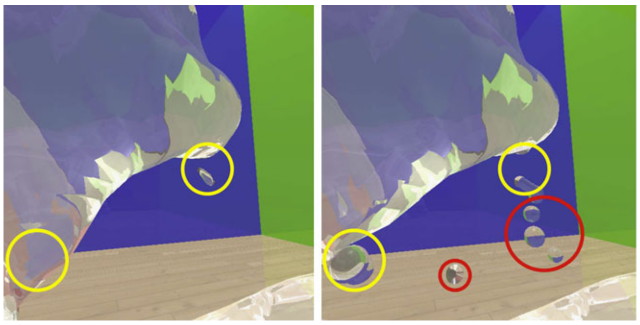
Fig. 5 Two snapshots of the bunny ear at the same frame. (*Left*) Without using metaballs. Some water drops do not appear due to the restricted grid resolution; (*right*) with metaballs (our methods). The *red and yellow circles* indicates the different parts between two snapshots

Comparison We compared our method with the method proposed in [7]. In [7], an ice particle is converted into some water particles when the ice particle absorbed enough energy. The ice particle is replaced by the water particles in the same place. Then interfacial interaction is performed between the water particles and the ice model. The water particles may move down along the surface. Our method gradually converts the ice particle into virtual water and the virtual water is transferred between ice particles. The virtual water contributes to the density field. Hence, the surface of the model is smooth in our method. Our method is capable of modeling the thin layer of the water flow on the surface of the model. The method in [7] requires densely sampled water particles for modeling the thin layer of water. In the real world, we observe that the ice on the floor melts and shrinks smoothly. We would not see a water droplet appears directly on the surface of the ice cube. Water usually emerges on the floor around the ice cube. We simulated this kind of phenomenon for melting an ice cube and compared their method with ours. In Fig. 6, the left two images show their result and the right two images show our result. We can see that their method creates water particles directly on the surface of the ice cube. Our method performs the transfer of virtual water via the ice particles and creates water particles on the floor around the ice cube.

Other examples We show three other examples: a bunny model on a table, five ice cubes and icicles in a cave. The

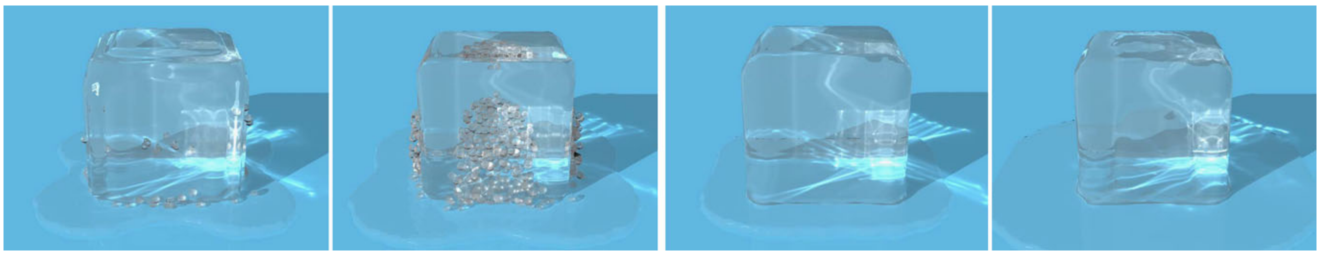

Fig. 6 (*Left two images*) The ice melting results proposed in [7]. If the size of the water particles is not appropriate, unnatural appearance occurs at the surface of the ice model. (*Right two images*) Our results. Our method converts the meltwater into the virtual water and water accumulates at the bottom of the ice model. The images were rendered by POV-Ray

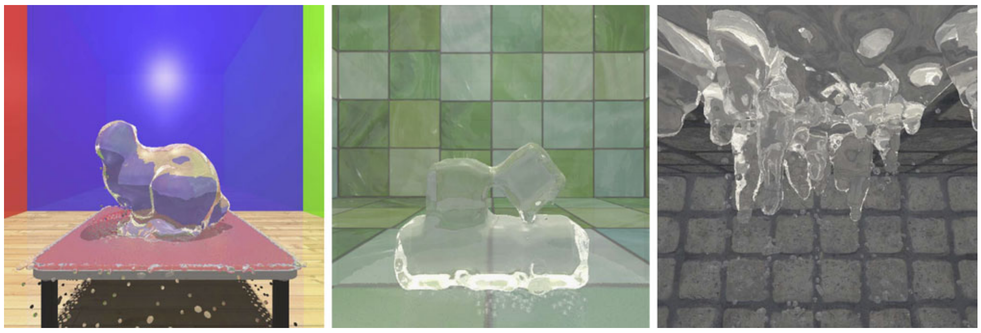

Fig. 7 (*Left*) A bunny on the table. The surface of the model shrinks smoothly while it melts. Water accumulates around the model. The water drops fall down at the boundary of the table. (*Middle*) Multiple ice cubes. The thin layer of water flows down and the water accumulates around the ice cubes. (*Right*) Icicles in the cave. Water drops fall off at the lower parts of icicles

snapshots of these three examples are shown in Fig. 7. The virtual water moves downwards and the water drops are formed at the lower parts of the models.

Limitations The limitations of our method are stated as follows. Our method simulates water flow as the transfer of virtual water between external ice particles. Thus, our method cannot simulate the detailed movement of water flows on the surface of ice models. We do not consider the external forces exerting on the ice particles, such as gravitational force. If the bottom part of the ice model melts, the ice model or parts of the ice model may float in the air. The affinity to water is high for the ice surface. Hence, when a water drop touches the ice surface, its shape is flatten in a very short time; and then it becomes a thin layer of water and it flows on the surface. Our method can simulate this kind of phenomenon. However, if water drops are in the air, their shape may not be spherical in real life. But our method employs metaballs for rendering water drops as spheres.

7 Conclusions and future work

We propose to incorporate an attribute called virtual water for each ice particle. The amount of virtual water of a particle indicates the amount of water around the ice particle. To simulate the water flow, we transfer the virtual water between the external ice particles. The virtual water of ice particles is converted into water particles for simulating water drops. The density field is defined by the ice particles and the virtual water. Our experimental results show that ice models shrink smoothly while they melt into water.

In the future, we would like to incorporate a force model into our system so that a part of the ice model falls down if the part is too heavy to be supported. Ice contains air bubbles due to the dissolved air in water [11]. Air bubbles are created at the ice–water interface according to the Stefan conditions [3] while the water is frozen. To realistically simulate ice melting, it would be interesting for melting ice with air bubbles.

Acknowledgements This work was supported by the National Science Council of ROC (Taiwan) (NSC-101-2221-E-009-157).

Appendix A: Density function

The density function $\Phi(\mathbf{x})$ is defined in the form

$$\Phi(\mathbf{x}) = \sqrt{\sum_a \left(f(X_a)\left(1 - \frac{r}{h}\right)\right)^2}, \tag{5}$$

where $r = |\mathbf{x} - \mathbf{x}_a|$, h is the particle interactive radius, X_a is the total volume of ice particle a (i.e. $X_a = V_a + Z_a$); and f is a function of volume and $f(V_{\text{init}}) = 1$. Consider a single ice particle and its total volume is X. The threshold is set to θ. Then the isosurface is a sphere and its radius R is computed as $R = (1 - \frac{\theta}{f(X)})h$. Based on the volume ratio of two spheres with and without virtual water, we should have

$$\frac{\frac{4}{3}(1 - \frac{\theta}{f(X)})^3 h^3 \pi}{\frac{4}{3}(1 - \frac{\theta}{f(V_{\text{init}})})^3 h^3 \pi} = \frac{V}{V_{\text{init}}}. \tag{6}$$

We rearrange the terms to obtain $f(X)$:

$$f(X) = \left(1 - \sqrt[3]{\frac{X}{V_{\text{init}}}}(1 - \theta)\right)^{-1} \theta. \tag{7}$$

For X close to V_{init} and $\theta = 0.5$, $f(X) \approx \sqrt[3]{\frac{X}{V_{\text{init}}}}$. This can be shown by using Taylor expansion.

References

1. Arash, O.E., Génevaux, O., Habibi, A., Dischler, J.M.: Simulating fluid–solid interaction. In: Graphics Interface, pp. 31–38 (2003)
2. Clavet, S., Beaudoin, P., Poulin, P.: Particle-based viscoelastic fluid simulation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 219–228 (2005)
3. Chen, S., Merriman, B., Osher, S., Smereka, P.: A simple level set method for solving Stefan problems. J. Comput. Phys. **135**, 8–29 (1997)
4. Fujishiro, I., Aoki, E.: Volume graphics modeling of ice thawing. In: Volume Graphics, pp. 69–80 (2001)
5. Foster, N., Fedkiw, R.: Practical animation of liquids. In: ACM SIGGRAPH, pp. 23–30 (2001)
6. Fujisawa, M., Miura, K.T.: Animation of ice melting phenomenon based on thermodynamics with thermal radiation. In: GRAPHITE, pp. 249–256 (2007)
7. Iwasaki, K., Uchida, H., Dobashi, Y., Nishita, T.: Fast particle-based visual simulation of ice melting. Comput. Graph. Forum **29**, 2215–2223 (2010)
8. Kanamori, Y., Szego, Z., Nishita, T.: GPU-based fast ray casting for a large number of metaballs. Comput. Graph. Forum vol. **27**, 351–360 (2008)
9. Lauterbach, C., Garland, M., Sengupta, S., Luebke, D., Manocha, D.: Fast BVH construction on GPUs. In: Comput. Graph. Forum **28**, 375–384 (2009)
10. Losasso, F., Irving, G., Guendelman, F., Fedkiw, R.: Melting and burning solids into liquids and gases. IEEE Trans. Vis. Comput. Graph. **12**(3), 343–352 (2006)
11. Madrazo, C., Tsuchiya, T., Sawano, H., Koyanagi, K.: Air bubbles in ice by simulating freezing phenomenon. J. Soc. Art Sci. **8**(1), 35–42 (2009)
12. Nishita, T., Nakamae, E.: A method for displaying metaballs by using Bézier clipping. Comput. Graph. Forum **13**, 271–280 (1994)
13. Nooruddin, F., Turk, G.: Simplification and repair of polygonal models using volumetric techniques. IEEE Trans. Vis. Comput. Graph. **2**, 191–205 (2003)
14. Solenthaler, B., Schläfli, J., Pajarola, R.: A unified particle model for fluid-solid interactions. Comput. Animat. Virtual Worlds **18**, 69–82 (2007)
15. Terzopoulos, D., Platt, J., Fleischer, K.: Heating and melting deformable models. J. Vis. Comput. Animat. **2**(2), 68–73 (1991)
16. Wojtan, C., Carlson, M., Mucha, P.J., Turk, G.: Animating corrosion and erosion. In: EurographicsWorkshop on Natural Phenomena, pp. 15–22 (2007)
17. Zhao, Y., Wang, L., Qiu, F., Kaufman, A., Mueller, K.: Melting and flowing in multiphase environment. In: Compute and Graphics, vol. 30, pp. 519–528 (2006)

Shing-Yeu Lii received his M.S. degree in Computer Science from National Chiao Tung University, Taiwan, ROC, in 2012. His research interests are computer graphics, computer animation and real-time rendering techniques.

Sai-Keung Wong received his M.Phil. and Ph.D. degrees in Computer Science from Hong Kong University of Science and Technology (HKUST) in 1999 and 2005, respectively. From June 2005 to January 2008, he was a postdoctoral fellow in Department of Computing at Hong Kong Polytechnic University. He has been an assistant professor of Computer Science Department at National Chiao Tung University, Taiwan, ROC, since 2008. His research interests include computer animation, collision detection, game engines and visualization.