Point-based Rendering of Water Surfaces with Splashes Simulated by Particle-based Simulation

Kei Iwasaki
Wakayama University

Kaori Ono
The University of Tokyo

Yoshinori Dobashi
Hokkaido University

Tomoyuki Nishita
The University of Tokyo

Abstract

Creating realistic animations of water is a challenging problem in the field of computer graphics. To create realistic animations of water, the fluid simulation of the movement of water is necessary. In recent years, attention has been paid to particle-based fluid simulation. Although several methods have been proposed to render the results of the particle-based simulation, there are few methods to render splashes. This paper presents an efficient rendering method for the results of the particle-based simulation with splashes. We calculate the water surfaces and the surfaces of the splashes by assigning the density to each particle and extracting iso-surfaces. The surfaces of the water and splashes are then sampled by points. This makes it possible to render the water surfaces and splashes efficiently by using point-based rendering.

1 Introduction

The research into fluid simulation is one of the most important research topics in computer graphics, so many methods have been developed to achieve this end [3, 4, 13]. Fluid simulations can be categorized into two types, grid-based simulation and particle-based simulation. Particle-based fluid simulations have been developed that represent the fluid as particles and calculate the fluid dynamics by solving the particles dynamics. Particle-based fluid simulations, such as Smoothed Particle Hydrodynamics (SPH) method [8] and Moving Particle Semi-Implicit (MPS) method [6], have several features compared to grid-based simulation. Particle-based simulations are free from the numerical diffusions in the advection term that represents the particle acceleration received from the velocity field. Moreover, particle-based simulations are suited for simulating topologically changes of water surfaces.

In recent years, point-based rendering methods have been developed, using a point as a primitive instead of polygons [1, 11]. In these methods, the object's surface is represented in a set of points. To represent continuous surfaces, an oriented disk (called *surfel*) is assigned to each point. Point-sampled objects are more flexible compared to polygons when it comes to handling highly complex or dynamically changing shapes, since they do not have to maintain globally consistent topological information. Therefore, rendering using point primitives is suited for particle-based simulation. Moreover, visualizing the results of particle-based fluid simulation by using point primitives is straightforward, since both of the result data of the simulation and the data from the rendering are unified into points.

In this paper, we render the results of the particle-based fluid simulation for water using point-based rendering. Since particle-based simulation is capable of handling the surface transformation easily, the possibility that the splashes are generated is high. Therefore, it is necessary to render splashes effectively. To figure out the motion of many small splashes, it must be simulated using many particles. However the calculation time is increased with respect to the number of particles. Thus, we propose an efficient rendering method taking into account splashes for the results simulated using relatively few particles (between several ten thousands and several hundred thousands particles).

The rest of our paper is organized as follows. In Section 2, previous work on rendering results of the fluid simulation is summarized. The overview of our method is described in Section 3. The representation of water surfaces and splashes is presented in Section 4. Rendering water surfaces and splashes is described in Section 5. Finally, results and conclusions are discussed in Sections 6 and 7, respectively.

2 Previous Work

There have been many methods for visualizing the results of the fluid simulation. The previous methods for rendering the results of the fluid simulation are categorized into two types. One is to polygonize the iso-surfaces, represented by implicit functions, and then render the polygons. Another is to directly render the implicit surface, without creating polygons. One of the methods to create the implicit surface using polygons is the marching cube method [7]. Many methods have employed the matching cube approach to render the water surface. Muller et al. [9, 10] proposed a particle-based simulation method and rendered the simulation result by using the marching cube method. Although the marching cube method is efficient, representing iso-surfaces by creating polygons requires the memory for the connectivity information.

Another visualization method for fluid simulation of water involves the rendering of the iso-surface directly. Enright et al. [2] and Premoze et al. [12] employed the level set method to represent the water surface. Their methods render the water surface by using Monte Carlo path tracing method. While these methods can render realistic images, the computational cost for the rendering is high. Moreover, representing small splashes is difficult by using these methods.

Takahashi et al. [14] generated the water surface by the marching cube approach and rendered the water surfaces by polygons. In addition, they generated small splashes and foam by their particle system and rendered them by quadrilateral polygons, called billboards, which textures of splashes and foams are mapped onto. Although they rendered very small splashes effectively, the representation of the changes of the shapes from the water surface to splashes is not enough, because the splash is rendered as a planar billboard.

In order to represent the realistic water surface, optical effects must be taken into consideration. We proposed a real-time method of rendering water surface taking into account optical effects [5]. We reconstructed the water surface from the results of particle-based simulation and rendered it taking into account caustics by using GPU. However, this method does not describe the rendering method of splashes and foam.

The proposed method renders the water surface taking into account the transformation of water surfaces into splashes by using point-based rendering.

3 Overview

Figure 1 shows the overview of our method. Our method deals with the results of the particle-based fluid simulation calculated by the MPS method (Figure 1(a)). To render the water surfaces, particles that represent the surfaces must be extracted. Direct rendering of the particles representing surfaces is one solution to visualize the results of the particle-based fluid simulation. However, the number of particles

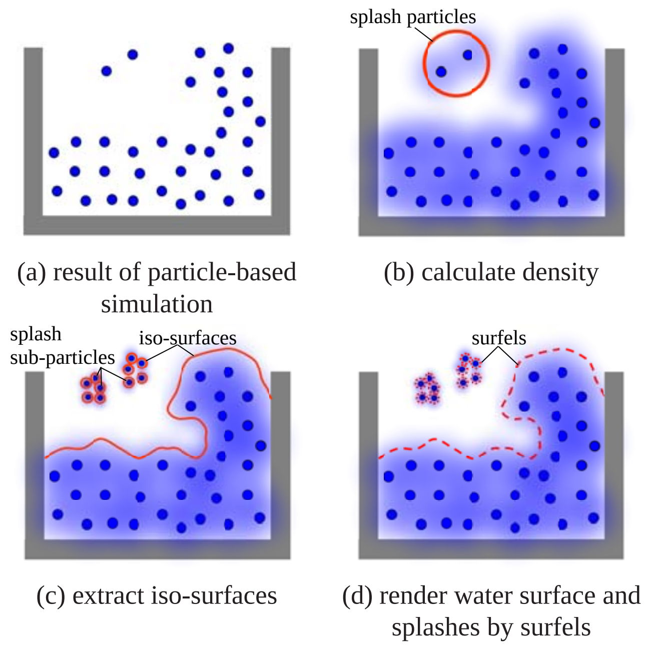


Figure 1. Overview of our method. (a) shows a result of the particle-based simulation. (b) We assign density to each particle and detect splash particles that represent splashes. Splash particles are subdivided into splash sub-particles. (c) We extract iso-surfaces representing water surfaces and surfaces of splashes. (d) We generate surfels representing water surfaces and surfaces of splashes. Finally, surfels are rendered.

used in the simulation is usually between about 1,000 and 1,000,000, so that the number of particles representing the surfaces is, at most, several ten thousands. On the other hand, point-based rendering methods are designed to render huge number of points measured by range scanners. Therefore, it is difficult to create high quality images of water surfaces by rendering only the particles used in the simulation.

To address this problem, our method generates dense sampling points on the water surfaces and surfaces of splashes. In our method, sampling points on the continuous water surfaces and those on surfaces of splashes are generated separately.

First, we assign a density distribution to each particle used in the simulation (Figure 1(b)). The density assigned to each particle decreases gradually toward the outside from the center of a particle. Then we detect the particles representing splashes. We define the particles, which used in the simulation and represent splashes, as *splash particles*.

Then we extract the iso-surface representing the water surface (Figure 1(c)). Extracted iso-surfaces are sampled by points. The sampling points on the water surface and surfaces of splashes are represented by surfels. We render the water surface and the surfaces of splashes by using surfels (Figure 1(d)). A normal vector of the surfel is calculated by a gradient vector of the water surface.

In addition, we render the water surface with splashes even when the number of particles used in the simulation is relatively small. In that case, a number of particles representing splashes is also small, so it is difficult to represent many small splashes only by rendering the iso-surface which is extracted from the density distribution of particles representing splashes. Therefore, we subdivide the splash particles into many small particles (Figure 1(c)). This achieves the effective representation of small splashes.

4 Representation of Water Surfaces and Splashes by Surfels

Each particle used in the simulation is assigned the density distribution defined by a density function. Then the densities of the particles are accumulated and the iso-surface is extracted in the simulation space. This gives us the water surface. The density function $W(r,h)$ is calculated from the following equation.

$$W(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \le r \le h \\ 0 & otherwise, \end{cases} \tag{1}$$

where r is the distance from the center of the particle to a calculation point, and h is the effective radius. In the particle-based simulation, each particle represents a mass of fluids. Initially, the particles are arranged at interval d. Therefore, we allocate volume d^3 to each particle. In this case, the density function W' at a point in the distance r from a particle is calculated by $W'(r,h) = W(r,h)d^3$. We represent the effective radius h as $h = nd$ where n is constant. In the MPS simulation, the effective radius h is selected as $2d \le h \le 4d$ in terms of the accuracy of the simulation. Therefore, we also set the constant n as $2 \le n \le 4$. The density function W' is expressed as the following.

$$W'(r,h) = \frac{315}{64\pi n^3}\left(1 - \left(\frac{r}{h}\right)^2\right)^3. \tag{2}$$

4.1 Representation of Splashes

In the particle-based simulation, splashes are often generated, since particle-based simulation is capable of tracking complex dynamic transformations of the fluid surfaces. To figure out the motion of many small splashes, a large number of particles are necessary in the simulation. However, the calculation time increases with respect to the number of the particles. To address this problem, we generate small splashes by making use of the results of the simulation using a relatively small number (between several ten thousands and several hundred thousands) of particles.

First, we detect the splash particles by the following method. Since a splash generally moves away from the water surface, the number of particles around a splash particle is considered to be small. This implies that the density at the center of the splash particle is likely to be small. Therefore, we identify a particle as the splash particle if the density at the particle position is lower than a specified threshold.

At each time step, we detect the particle that has just changed into the splash particle and then subdivide the splash particle. That is, the splash particle is replaced with many particles whose effective radius of the density distribution is smaller than the splash particle. We call the subdivided particles *splash sub-particles*. The effective radius of the splash sub-particle is determined so that the summation of the densities from the splash sub-particles is equal to the density from the original splash particle.

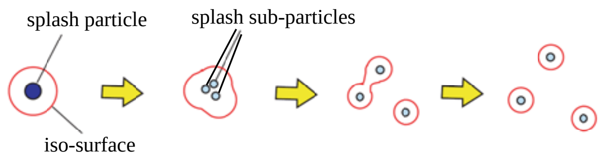


Figure 2. Subdivision of splash particle into splash sub-particles.

The effective radius of the splash sub-particle is calculated as follows. Initially, we assume that a splash particle is subdivided into V splash sub-particles. Each particle used in the simulation has a volume d^3 as described before. Therefore, the volume of a subdivided splash sub-particle is considered as d^3/V. The radius R_s of the sphere with the volume d^3/V, is calculated as the following equation.

$$R_s = \left(\frac{3}{4\pi V}\right)^{1/3} d = k_s d, \tag{3}$$

where k_s is $(3/4\pi V)^{1/3}$. Then, we express the small effective radius of the splash sub-particle h'. The density W_s'' that a sub-splash particle is assigned is calculate from the following equation.

$$W_s''(r,h') = \frac{315}{64\pi(h')^9}((h')^2 - r^2)^3 \frac{d^3}{V}. \tag{4}$$

We assume that the value of W' for extracting the continuous water surface is W_R. The effective radius of a splash sub-particle h' is determined so that $W''_s(R_s, h') = W_R$.

The subdivision of splash is caused by the resistance of the air. Therefore, the number of splash sub-particles depends on the velocity of the splash particle. In this paper, we generate several patterns of subdivisions of the splash particle. In the rendering process, we select the pattern corresponding to the maximum velocity of the original splash particle. Figure 3 shows a pattern of subdivisions of the splash particle. The motions of the splash sub-particles follow the motion of the original splash particle.

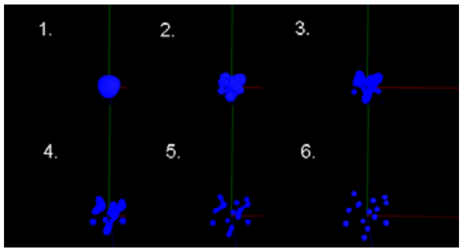


Figure 3. An example of patterns of subdivisions of a splash particle.

4.2 Extraction of iso-surfaces

The water surfaces are represented as the iso-surfaces extracted from the density distribution in the simulation space. In the extraction, we distinguish the surfaces of splashes existing discrete from water surfaces except the surfaces of splashes, and extract the iso-surfaces respectively.

To extract the water surfaces, we create a temporary grid in the simulation space. Then we calculate the density at each grid point and extract the sampling points of the iso-surfaces between the grid points. On the other hand, splashes are small compared to the interval of the grid and the iso-surfaces of the splashes cannot be extracted by using the grid. Therefore, we extract the iso-surfaces radially from the center of the splash sub-particles, and obtain the sampling points on the surfaces of splash sub-particles.

5 Rendering Water Surfaces and Splashes

We render the water surfaces and splashes represented by sampling points by using surfels. We calculate the color of surfels by taking into account reflection and refraction of light.

When we render the water surface, aliasing may occur if the color difference in the surfel is large. To address this, we subdivide the surfels where aliasing occurs. First, we calculate the colors at four points slightly away from the center of the surfel. If the color difference is larger than a threshold C_α, the surfel is subdivided into four surfels as shown in Figure 4. If the color difference is smaller than C_α, the surfel is rendered with the average color of four colors at four points. The surfel is adaptively subdivided until the difference of the internal four colors is less than the threshold.

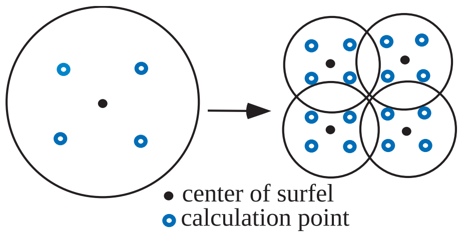
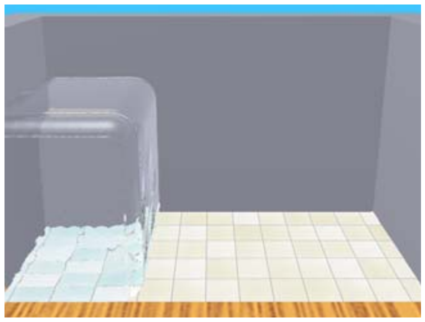
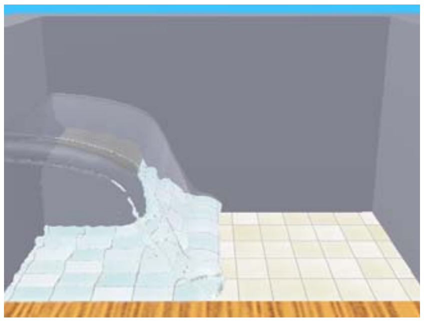


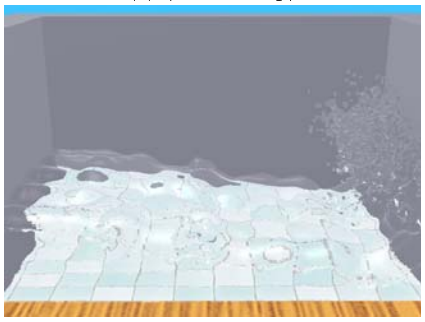
Figure 4. Adaptive surfel subdivision.

6 Results

Figure 5 shows the rendered results of water column collapse simulation. As shown in Figure 5, changes of complex shapes from continuous water surfaces to discrete small splashes can be rendered by using our method. Figure 6 shows the rendered results of simulation of dropping a rigid parallelepiped into the water. In the right side images of Figure 6, splash particles are subdivided into small splash sub-particles. On the other hand, the left side images of Figure 6, the splash particles are directly rendered without subdivision. Compared to Figures 6(a) and (c), Figures 6(b) and (d) give unnatural impressions. In Figures 6(b) and (d), a splash particle is subdivided into utmost 25 splash sub-particles. In Figure 5, the size of the aquarium is $1.23 \times 0.72 \times 0.57(m)$. The number of particles used in the simulation is about 7,300. In Figure 6, the size of the aquarium is $1.0 \times 1.0 \times 0.8(m)$ and the height of the water is $0.65(m)$, the specific gravity of dropped rigid is 1.5, and the initial position of the rigid is $0.8(m)$ from the bottom of the aquarium. Each time step in the simulation is set to $0.01[s]$ and the number of particles used for the simulation is about $210,000$. Figure 7 shows the result of the adaptive surfel subdivision. Figure 7(a) shows the image rendered with adaptive subdivision of surfels. Figure 7(b) shows the image rendered without subdivision. As shown in Figure 7, aliasing due to color difference is relaxed. The rendering time for Figure 6(c) is 17.8 seconds, and Figure 6(d) is 19.2 seconds on a standard PC with PentiumD 3.0GHz. As shown in these figures, our method can render realistic water surfaces and small splashes.


(a) (0 time step)


(b) (15 time step)


(c) (50 time step)

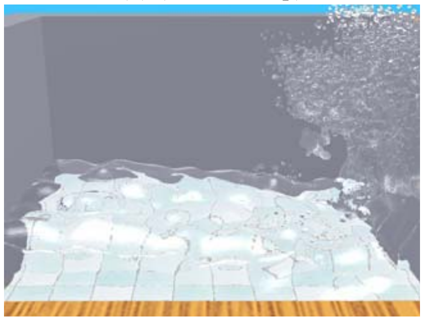
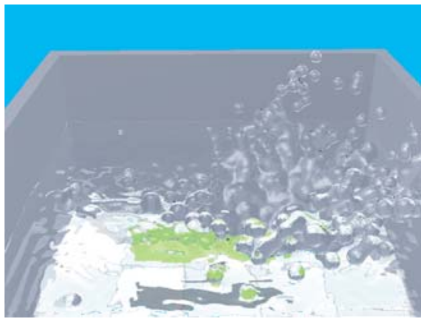
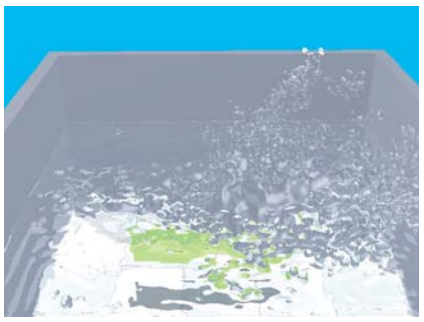
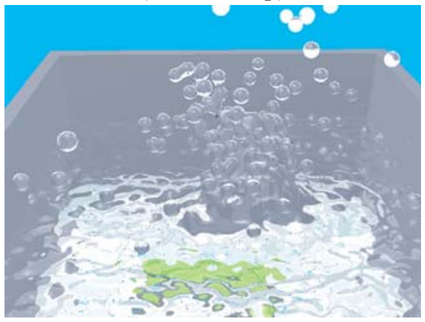

(d) (60 time step)

Figure 5. Examples of rendered results of water column collapse simulation.


(a) without subdivision
(50 time step)


(b) with subdivision
(50 time step)


(c) without subdivision
(70 time step)

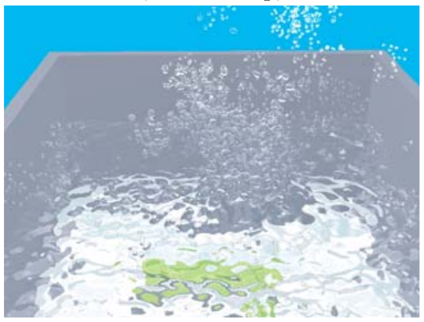

(d) with subdivision
(70 time step)

Figure 6. Examples of rendered results of simulation of dropping a rigid parallelpiped into water.

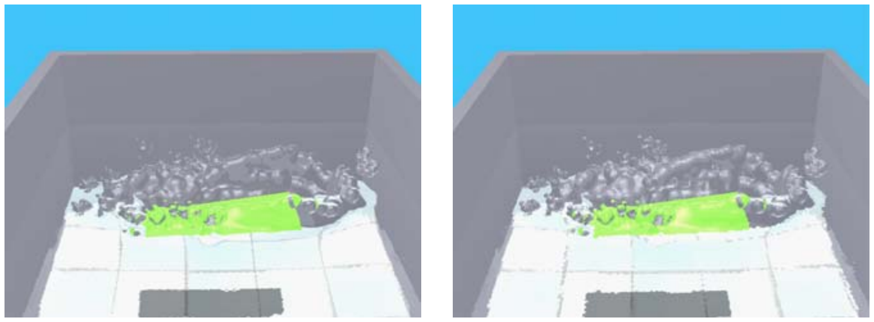

(a) with adaptive subdivision (b) without subdivision

Figure 7. Comparison between surfels with adaptive subdivision and without subdivision.

7 Conclusions

We have proposed a point-based rendering method for the results of the particle-based simulation, taking into account splashes. Our method can render small splashes from the result of the particle-based simulation using a relatively small number of particles. We extract dense samping points representing surfaces of water and splashes. This makes it possible to render the water surfaces and splashes by using point-based rendering.

In future work, we would like to accelerate the rendering using GPUs. We also would like to render foam to enhance the realism.

Acknowledgements

We would like to thank Prometec Software Inc. for providing the simulation data.

References

[1] C. Co, B. Hamann, K. Joy, *"Iso-splatting: A Point-based Alternative Isosurface Visualization"*, Proc. Pacific Graphics 2003, 2003, pp.325-334.

[2] D. Enright, S. Marschner, R. Fedkiw, *"Animation and Rendering of Complex Water Surfaces"*, Proc. SIGGRAPH 2002, 2002, pp.736-744.

[3] N. Foster, D. Metaxas, *"Realistic Animation of Liquids"*, Proc. Graphics Interface 96, 1996, pp.204-212.

[4] N. Foster, R. Fedkiw, *"Practical Animation of Liquids"*, Proc. SIGGRAPH 2001, 2001, pp.23-30.

[5] K. Iwasaki, Y. Dobashi, F. Yoshimoto, T. Nishita, *"Real-time Rendering of Point-based Water Surfaces"*, Proc. Computer Graphics International 2006, to appear.

[6] S. Koshizuka, H. Tamako, Y. Oka, *"A Particle Method for Incompressible Viscous Flow with Fluid Fragmentation"*, Computational Fluid Dynamics Journal, Vol.29, No.4, 1995, pp.29-46.

[7] W. Lorensen, H. Cline, *"Marching Cubes: A high Resolution 3D Surface Construction Algorithm"*, Proc. SIGGRAPH'87, 1987, pp.163-169.

[8] L.B. Lucy, *"A Numerical Approach to The Testing of the Fission Hypothesis"*, The Astronomical Journal, Vol.82, No.12, 1977, pp.1013-1024.

[9] M. Muller, D. Charypar, M. Gross, *"Particle-based Fluid Simulation for Interactive Applications"*, Proc. Symposium on Computer Animation 2003, 2003, pp.154-159.

[10] M. Muller, B. Solenthaler, R. Keiser, M. Gross, *"Particle-Based Fluid-Fluid Interaction*, Proc. Symposium on Computer Animation 2005, 2005, pp.237-244.

[11] H. Pfister, M. Zwicker, J. Baar, M. Gross, *"Surfels: Surface Elements as Rendering Primitives"*, Proc. SIGGRAPH 2000, 2000, pp.335-342.

[12] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, R.T. Whitaker, *"Particle Based Simulation of Fluids"*, Computer Graphics Forum, Vol. 22, No. 3, 2003, pp.335-343.

[13] J. Stam, *"Stable fluids"*, Proc. SIGGRAPH'99. 1999, pp.121-128.

[14] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, H. Ueki, *"Realistic Animation of Fluid with Splash and Foam"*, Computer Graphics Forum, Vol. 22, No. 3, 2003, pp.391-400.