Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Practical Animation of Turbulent Splashing Water

Janghee Kim[1] Deukhyun Cha[2] Byungjoon Chang[2] Bonki Koo[1] Insung Ihm[2]

[1] Electronics and Telecommunications Research Institute, Daejeon, Korea
[2] Sogang University, Seoul, Korea

Abstract

Despite recent advances in fluid animation, producing small-scale detail of turbulent water still remains challenging. In this paper, we extend the well-accepted particle level set method in an attempt to integrate the dynamic behavior of splashing water easily into a fluid animation system. Massless marker particles that still escape from the main body of water, in spite of the level set correction, are transformed into water particles to represent subcell-level features that are hard to capture with a limited grid resolution. These physical particles are then moved in the air through a particle simulation system that, combined with the level set, creates realistic turbulent splashing. In the rendering stage, the particle's physical properties such as mass and velocity are exploited to generate a natural appearance of water droplets and spray. In order to visualize the hybrid water, represented in both level set and water particles, we also extend a Monte Carlo ray tracer so that the particle agglomerates are smoothed, thickened, if necessary, and rendered efficiently. The effectiveness of the presented technique is demonstrated with several examples of pictures and animations.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Physically based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Animation

1. Introduction

1.1. Problem specification

As a consequence of recent research efforts into the physically based animation of water-like liquids, nowadays photorealistic water effects are routinely generated in the animation production industry. Among the successful simulation techniques devised by the computer graphics community, the hybrid particle level set method, combined with an octree data structure, produces highly realistic water behavior, by adaptively solving the full three-dimensional Navier–Stokes equations [LGF04].

Despite its excellent results, however, such a method, in which water is represented on an Eulerian grid, often suffers from a difficulty in depicting the detailed turbulent behavior of splashing water, involving the complicated formation of water spray, wave breaking, mist, foam, and complex free surface motion. Using more three-dimensional grid cells would supply more convincing results for the Eulerian approach, but their number increases cubically with respect to the resolution adopted, quickly leading to impractical requirements for computation time and memory space. In fact, these natural phenomena are known to be among the most poorly understood fields in fluid dynamics.

In parallel with the grid-based approaches, meshless simulation methods have been also developed for water animation, where particles are employed to keep track of the free surface and to solve the underlying partial differential equations. Recently introduced particle-based techniques have shown that these Lagrangian approaches offer a computationally simple and versatile simulation scheme, although reconstructing a natural-looking smooth surface from simulated particles is often difficult.

Particles are an excellent means of producing the dynamic and fuzzy effects of natural phenomena when conventional modeling techniques do not work. They are well suited to producing the small-scale detail of turbulent splashing water that is hard to capture with an Eulerian grid of limited resolution. The primary goal of this work is to combine both grid-based and particle-based simulation techniques to integrate easily the automatic production of turbulent water effects into a fluid animation system. In particular, we extend the well-accepted particle level set method in an attempt to complement the grid-based method, which suffers from a

© The Eurographics Association 2006.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

difficulty in representing subcell-level features such as tiny droplets and spray. We demonstrate that combining both the Eulerian and the Lagrangian techniques greatly improves the visual reality of splashing effects in turbulent water.

1.2. Previous work

In computer animation, the work of Foster and Metaxas [FM96] is probably the first that attempted to solve the full three-dimensional equations to produce realistic water effects. After some years, Foster and Fedkiw [FF01] improved the water simulation scheme significantly by adopting a semi-Lagrangian integration scheme (which was introduced to the graphics community by Stam [Sta99]) to solve stably for liquid motion. In modeling the water surface, they employed a hybrid front tracking model, combining a dynamic level set and massless marker particles, where the particles are used to reconstruct the level set surface in regions that are poorly resolved.

Enright et al. [EMF02, ELF05] presented an enhanced hybrid front-tracking method, called the particle level set method, which exhibits significantly more realistic behavior for a complex water surface. Combined with an octree-based adaptive computation model [LGF04], this water simulation technique is able to generate very impressive water animation, capturing fine-scale features of the water surface.

In parallel with these efforts, the incompressible Navier–Stokes equations were also solved for water animation using a variety of extensions and modifications: Hong and Kim [HK03] introduced a volume-of-fluids algorithm for building a multiphase fluid simulator that handles both water and air, mainly focusing on buoyancy and surface tension in their animation of bubbles. A moving grid windowing method was proposed by Rasmussen et al. [REN*04] to track only the visually interesting portion of a liquid animation. Song et al. [SSK05] also attempted to suppress dissipation, arising from the semi-Lagrangian method, by adopting Constrained Interpolation Profile (CIP)-based advection, and converting potentially dissipative cells into droplets or bubbles. Recently, Hong and Kim [HK05] proposed a numerical technique that effectively handles the discontinuities in physical properties at interfaces between different fluids.

Along with these grid-based methods, many different particle-based techniques from the field of Computational Fluid Dynamics (CFD) have been also adapted for liquid animation. A particle-based fluid simulation was introduced to the graphics community by Stam and Fium [SF95], where a set of "fuzzy" particles were used to solve diffusion equations, in the hope of reducing the severe memory overheads for the grid-based method. Desbrun and Gascuel [DG96] applied the Smoothed Particle Hydrodynamics (SPH) method [Luc77, GM77] to the animation of viscous fluids with a wide range of stiffness and viscosity.

Recently, the SPH technique was applied by Müller et al. [MCG03], focusing on the interactive simulation of water. In subsequent work, Müller et al. [MSKG05] extended the SPH-based simulation model to consider fluid–fluid interaction, enabling it to handle such phenomena as multiple fluids, trapped air, and phase transition. Premože et al. [PTB*03] simulated water, using the Moving-Particle Semi-Implicit (MPS) method [KTO95]. This alleviates the difficulty of enforcing the incompressibility condition, arising from the SPH technique. Zhu and Bridson [ZB05] also presented a particle simulation system appropriate for animating sand and water, based on two hybrid fluid simulation methods called Particle-In-Cell (PIC) [Har64] and Fluid-Implicit-Particle (FLIP) [BR86]. In animating viscoelastic fluids, Clavet et al. [CBP05] exploited particles whose positions were updated using a method called double density relaxation, to enforce incompressibility and particle anti-clustering.

In addition to the works cited above, Lagrangian particles have also been used, often complementing the grid-based approaches, with the aim of modeling fine details of water that are difficult to create with the underlying grid resolution. Sims [Sim90] and O'Brien and Hodgins [OH95] used a particle system to model water droplets and splashes, in which simple dynamics equations were applied to model particles falling under gravity, without considering interaction between them. Foster and Fedkiw [FF01] directly rendered the particles that are escaped despite the level set correction in regions of high curvature, treating them as small water drops. Takahashi et al. [TFK*03] combined a particle system with a CIP-based fluid system that solves the full Navier–Stokes equations, where particles intending to represent splashes and foams were created in a high-curvature water surface, and advected independently under the influence of velocity fields, obtained through a volume-of-fluid technique and/or gravity. Greenwood and House [GH04] generated the visual effect of bubbles, based on the method of Enright et al. [EMF02], by creating them whenever air marker particles moved too far across the fluid's interface, and became trapped inside the fluid. On the other hand, Guendelman et al. [GSLF05] rendered the negative particles, removed from their particle level set system, to depict splash and spray. Selle et al. [SRF05] generated turbulent water effects by seeding vorticity-carrying particles in the flows to overcome the weakness of the semi-Lagrangian scheme.

1.3. Our contribution

In this paper, we present a practical animation technique that easily generates a natural appearance of turbulent splashes in a particle-level-set-based liquid animation system. In developing our scheme, we aim to minimize the extra cost induced by the extension. For this, the massless marker particles that still escape from the main body of water, despite the error correction, are used as seeds to produce the subcell-level detail. In contrast to previous works that also exploit the stray particles [FF01,GH04,GSLF05] or the particles generated in high-curvatured regions [TFK*03], for generating droplets and bubbles, our technique attempts to estimate the volume loss quantitatively and distributes it to water particles. These physical particles are then moved in the air by an advanced particle simulation system that, combined with the dynamic level set, creates realistic behavior for a turbulent splash. We show that their physical properties, such as mass and velocity, may be used effectively to generate the natural appearance of the tiny water agglomerates and spray that are induced by the turbulent movement of water.

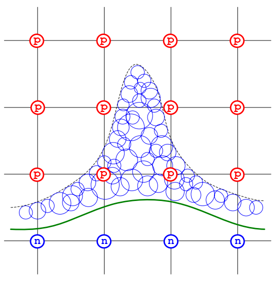
We also present rendering techniques suitable for taking a close-up view of merging water drops with minimized aliases. These smoothing and thickening methods, easily integrated into a Monte Carlo ray tracer, enable an animator to control the small-scale rendering effects for creating turbulent splashes. Our experiments with several example scenes demonstrates that the interesting subcell-level features of splashing water may be added to the level-set-based simulation system at a very small additional cost, and that the presented visualization techniques easily create realistic visual effects.

2. Modeling of water particles

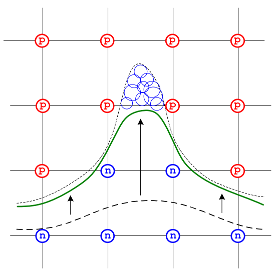
2.1. Identification of underresolved regions

The presented hybrid simulation system is built with a level set simulator that represents the main body of splashing water, and a particle simulator that depicts its small-scale detail. For the level set evolution, we use the particle level set method, with $\phi \leq 0$ representing water and $\phi > 0$ designating air. Combined with an octree data structure, the level set simulator creates very detailed surfaces of dynamically evolving water, although it is not easy to generate splashing effects such as tiny droplets, spray, and mist.

The primary idea of the hybrid particle level set method is to rebuild the level set in underresolved regions using massless marker particles. Its effectiveness has been demonstrated in a variety of liquid animation systems. However, it sometimes fails to recover subcell-level features, which often occurs when water undergoes turbulent motion. Figure 1 illustrates such a case of small-scale detail being lost. In (a), the escaped negative marker particles (blue circles) are shown with the water interface (green curve) that has been evolved with a level set method only. Through the error correction process of the particle level set method, the interface is forced to move upwards, reducing its local volume loss as shown in (b). Unfortunately, there still remain escaped negative particles near the sharp corner, even after the correction stage. This phenomenon is due to the fact that the stretching part is too tiny to be resolved with the current grid resolution.



(a) Before correction



(b) After correction

Figure 1: *Marker particles that are still missing after the error correction process. The circled characters indicate the sign of the level set function at grid points. The particle level set method rebuilds the interface (green curve), exploiting the error information provided by the escaped negative maker particles (blue circles). This hybrid technique offers quite an effective interface-capturing scheme. However, it still suffers from the difficulty of reconstructing subcell-level features.*

In the original particle level set method [EFFM02], the missing marker particles are kept alive in the subsequent simulation, in the hope that they agglomerate and contribute to the level set in the future. In fact, the missing particles provide an excellent clue to the generation of various rendering effects for splashing water. Rather than wait until they reappear in the level set, we explicitly transform these *marker* particles into *physical* particles to model small-scale detail that the given Eulerian grid fails to represent.

2.2. Generation of water particles

In the work of Song et al. [SSK05], a smeared Heaviside function was applied to potentially dissipative cells in order to detect small isolated regions, and to convert them into droplets or bubbles with an estimated volume. This first-order approximation tool may be an effective means of generating visually pleasing effects in cases where there is no information other than level set values to imply any subcell-level features.

On the other hand, the particle level set method offers more information that helps correct poorly resolved regions of the flow. The maker particles that move with a higher-order accuracy enables the extraction of more characteristic information about the dissipating water. Recall that the purpose of using marker particles in the particle level set method is to improve interface capturing, not to represent a specific volume of water. A side effect of the marker particles is that their agglomerates, formed by overlapping escaped negative particles, offer a good indication of the volume and shape of the water lump that still dissipates despite the error correction process.

In order to exploit fully the information provided by the particle level set method, we utilize the lost particles to treat subcell-level details. In our implementation, a negative particle is defined as escaped when it crosses the corrected interface by more than its radius. Then the volume that is lost in a cell is approximated by a Monte Carlo integration technique. For every cell that contains at least one escaped particle, a series of random three-dimensional points are repeatedly generated within the cell, checking if they fall in any of the escaped marker particles. The ratio of inner points, multiplied by the cell's volume, then gives a good approximation to the actual minute volume of disappearing water.

Once the volume loss in a cell is estimated, the escaped maker particles are transformed into physical water particles

(unit: %)

	(0.0, 1.0]	(1.0, 2.0]	(2.0, 3.0]	(3.0, 4.0]	(4.0, 5.0]	(5.0, 6.0]	(6.0, ∞)
Scene 1	61.8	18.7	9.0	4.5	2.6	1.4	2.0
Scene 3	56.3	21.6	10.0	4.8	2.8	1.6	2.9

Table 1: *Average ratios of volume loss in a cell. A simple Monte Carlo integration method was applied to the estimation of volume loss that arises despite the error correction of the particle level set method. Two animation scenes, displayed in Figure 4(a) and (c), were tested, where the ratios for the cells containing at least one escaped marker particle were averaged over the first 600 simulation frames. Here, the value in column $(a,b]$ represents the ratio of volume-losing cells whose volume loss ratio falls in the interval $(a\%, b\%]$. As shown in the table, most volume-losing cells lose less than 3% of volume. The statistics, in fact, reveal that the particle level set method traces the error-prone level set very well, while the small amount of volume loss may be exploited for depicting the small-scale detail of splashing water, complementing the grid-based method.*

in such a way that the estimated volume is distributed to the water particles. Assume that there exist n_p escaped marker particles in the cell whose cluster takes up volume V inside it. Each water particle inherits the position and velocity from the corresponding marker particle. Then its radius is initially set to the average radius $r_{ave} = \sqrt[3]{\frac{3V}{4\pi n_p}}$. To make a more natural rendering effect, it is also possible to distribute the initial radii according to, for example, a Gaussian distribution, so that the total sum becomes V. Once the radius of each particle is determined, by either method, the water particles are examined to see if any two particles overlap excessively, incurring a noticeable error in volume reconstruction. If this occurs, they are merged into a single particle with a radius $r_{new} = \sqrt[3]{r_1^3 + r_2^3}$, where r_1 and r_2 are the radii of the overlapping particles. Once generated, the water particles are added into the particle simulation system, and then undergo Lagrangian motions until they are absorbed back into the main body of water or leave the scene.

Table 1 shows a typical distribution of estimated ratios of the volume loss that occurs after the error correction stage. In this experiment, we sampled 2048 times per volume-losing cell for a precise approximation, while our experience indicates that 256 samplings per cell are usually enough to obtain reliable volume-loss estimates. Note that the additional overhead caused by the integration process is not severe because it is carried out only in those cells that contain an escaped marker particle. As revealed in the statistics, the ratios are usually quite low. In fact, this experimental result demonstrates that the particle level set method performs very well in rebuilding the level set in underresolved regions. On the other hand, the small amount of volume loss explains the induced errors when an Eulerian grid of limited resolution is used. By representing the lost volume (even though very small) as water particles, we are enabling the effective generation of detailed splashing effects whose depiction is difficult with grid-based simulation methods.

2.3. Advection of water particles

In order to produce visually convincing splashing effects, it is important to implement an appropriate simulator for advecting water particles. In the computational fluid dynamics community, several meshfree particle methods or particle-like methods have been developed and extended in order to have a range of mixed features. Among them, the SPH method and its variants are widely used for discrete particle systems. Although they work well in general, our preliminary implementation indicates that the water particles tend to scatter excessively when animating turbulent water. We assume that this phenomenon arises from the difficulty in enforcing the incompressibility constraint, especially for nonuniformly distributed particles.

While such extensions as the MPS method, already applied in the computer graphics community [PTB*03], may possibly be exploited, we find that combining the PIC [Har64] and FLIP [BR86] methods, both recently introduced to the graphics community by Zhu and Bridson [ZB05], leads to an effective implementation that easily produces visually pleasing behavior for turbulent splashes. One advantage of this hybrid implementation is that it is possible to tune the viscosity of splashes, which permits control of the smoothness of surfaces. As in [ZB05], we use a weighted average of PIC and FLIP when particle velocities are updated after the nonadvection steps. On the other hand, when the particle velocities are transferred to an auxiliary Eulerian grid, we extend the standard trilinear interpolation so that the particle's mass is also reflected as a weighting factor. Since the mass is distributed nonuniformly to particles in our scheme, this modification will reflect the influence of the water particles more precisely.

In coding the hybrid particle system, we usually set the resolution of the auxiliary grid to the same effective resolution as the grid used in the adaptive level set computation, although they do not have to align. Note that this secondary grid is dynamically allocated only in the regions where the water particles exist. Since they usually take up only a small portion of the computational domain, a simple memory management technique enables efficient implementation without a great increase in memory requirements.

2.4. Interaction of water particles with level set surface and objects

The interaction of water particles with themselves is reflected in two ways. First, any two particles overlapping too much are merged into a single particle in the water particle generation step. Secondly, the masses and velocities of water

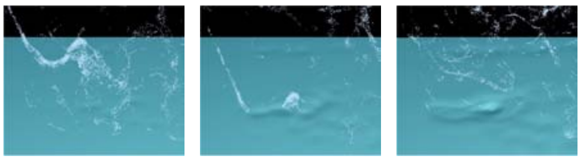


Figure 2: *Impact of falling droplets on water surface. The individual masses of the particles are minuscule. However, their impact accumulates as the droplets continue to hit the surface, causing noticeable ripples.*

particles are used in the PIC/FLIP-based particle simulator, that causes the particles to interact to each other.

On the other hand, their interaction with the main body of water and objects must be carefully coded into the simulation system. When a water particle hits the level set interface during its advection, it becomes absorbed into the main body of water. As the returning water particle has mass, the velocity field of the Eulerian grid representing the level set is affected near the hit point. Let m_p and $\mathbf{v}_p$ be the mass and velocity of the particle, respectively, at the moment of collision. Let m_{ijk} and $\mathbf{v}_{ijk}$ denote the corresponding quantities for the surface cell hit by the particle. Then, according to the law of momentum conservation, the new velocity of the cell is $\mathbf{v}_{ijk}^{new} = \frac{m_p \cdot \mathbf{v}_p + m_{ijk} \cdot \mathbf{v}_{ijk}}{m_p + m_{ijk}}$. As the density is constant, this can be expressed as $\mathbf{v}_{ijk}^{new} = \frac{V_p \cdot \mathbf{v}_p + V_{ijk} \cdot \mathbf{v}_{ijk}}{V_p + V_{ijk}}$ in terms of volume quantities, where the volumes V_p and V_{ijk} can be approximated by using the particle's radius, and the technique described in Subsection 2.2, respectively (see Figure 2 for the particle-level set interaction).

In addition to the velocity, the volume of the cell needs be updated properly. To achieve this, level set values in the neighboring cells need to be updated, by solving the inverse problem to that of computing of the volume of particles that depart from the main body. This is not a cheap computation because it entails the adjustment of marker particles to the new interface as well as the modification of level set values.

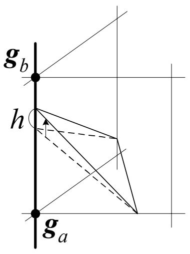


Figure 3: *Approximation of volume increase*

In [SSK05], the inverse smeared Heaviside function was applied to modify the neighboring level set values. Another possible, simple-to-implement update method is to find the edge of grid, closest to the returning water particle, that intersects with the level set interface, and adjust the level set values at its two incident grid points (see the edge connecting $\mathbf{g}_a$ and $\mathbf{g}_b$ in Figure 3). Now, their level set values are to be increased by h so that the intersection point between the interface and the edge moves upward by distance h. Then the entire volume increase induced by this modification can be approximated by summing the volumes of all incident tetrahedra, formed by the triangles of iso-surface obtained using the trilinear interpolation and the marching cube algorithm (the triangle in dashed line), and the new triangles formed as a result of moving the intersection point (the triangle in solid line). Since the volume of a tetrahedron is one-third the distance from a vertex to the opposite face, times the area of that face, it is possible to express the total volume increase in a linear expression of h. As we know the volume of the returning water particle V_p, the unknown h may be easily computed.

Note that the actual volume of absorbed water particles is quite small compared to that of the main body of water (refer to Table 2 in Section 4). While the minute volume of re-entering particles can be added back to the main body in level set form using either of the mentioned techniques, this process may often be omitted without making a visual difference in the animation results.

In addition to the main body, the water particles also interact with solid objects in the environment. When a particle hits an object with elasticity coefficient α_e, it bounces back from the surface with the velocity $\mathbf{v}_p^{new} = -\alpha_e \cdot \mathbf{v}_n + \mathbf{v}_t$, where $\mathbf{v}_n$ and $\mathbf{v}_t$ are the normal and the tangential velocities, respectively, of the particle at the moment of the collision. By introducing random noise to α_e and $\mathbf{v}_t$, the splashing water spatters on solid boundaries more naturally. See Figure 4 for some simulation results.

3. Rendering of a close-up view of water particles

Most particle rendering systems, including the classical technique presented by Sims [Sim90], draw particles point-by-point without considering their connectivity, which is usually satisfactory when they are seen from a distance. When the splashing water is viewed closely, however, it is often desirable to reconstruct complex surfaces of tiny water drops. However, this is difficult to achieve with sphere-based particle clusters, due to problems in controlling the resulting smoothness of the extracted surfaces. Insufficient smoothing produces rough-looking bumps. On the other hand, too much smoothing blurs the small-scale detail that a particle-based simulation technique needs to generate. In this section, we present two useful techniques that enable an animator to control the small-scale rendering effects for creating turbulent splashes. Then, we also describe how to generate water spray in our ray tracer. Note that the smoothing and thickening techniques are unnecessary when scenes are taken, like the examples in Figure 8, from some distance. These supplementary rendering tools are just for reducing visual aliases that may occur when taking a very close-up view of particle clusters.

3.1. Smoothing particle clusters

Several surface reconstruction techniques for particle-related water simulations have been proposed, including those in [FF01, MCG03, PTB*03, ZB05], which employ an Eulerian grid to extract a set of isosurfaces from a scalar

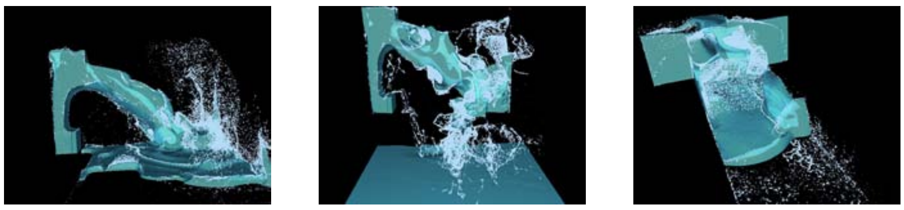

(a) Scene 1 (b) Scene 2 (c) Scene 3

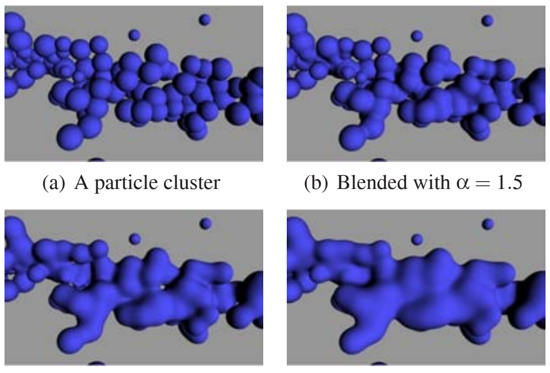
Figure 4: *Simulation results. These snapshot images are taken from three different test scenes, where both level sets and particles are displayed. The corresponding rendering images are shown in Figure 8.*

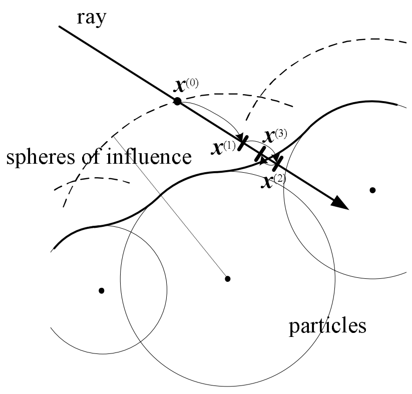
field defined by the particles. When particle clusters are visualized in detail, it is undesirable to generate their polygonal models explicitly, because a memory-consuming high resolution grid must be used. Instead of adopting such an approach, we have designed a smoothing technique for tiny particle clusters that permits direct control over the smoothness of reconstructed water surfaces. As will be described below, our smoothing scheme, similar to the blending techniques of other works [Bli82, WMW86], is naturally combined with the level-set-based ray tracer without any need to generate intermediate isosurfaces.

Consider a set of particles $\mathbf{p}_i = (\mathbf{x}_i, r_i)$ with center $\mathbf{x}_i = (x_i \;\; y_i \;\; z_i)^t$ and radius r_i. Given a point $\mathbf{x} \in R^3$, we define a scalar function $W(\mathbf{x})$ as a weighted sum of contributions from nearby particles: $W(\mathbf{x}) = \sum_i W_i(\mathbf{x})$. Here, $W_i(\mathbf{x})$ is a smoothing kernel whose detail is provided in the Appendix. For a pair of global parameters α (> 1) and $\beta \in (0,1)$, it has some interesting properties. First, for a particle i, its radius of influence is $\alpha \cdot r_i$. Furthermore, its spherical boundary can be extracted from the implicit equation $W_i(\mathbf{x}) = \beta$.

By summing the smoothing kernels, and finding the isosurface corresponding to $W(\mathbf{x}) = \beta$, we can blend particle clusters quite easily. Figure 5 shows an example of cluster smoothing with different α values: (a) displays an original particle cluster represented in terms of spheres. The other figures show the smoothed surfaces with $\alpha = 1.5$, 2.0, and 2.5, respectively. Our experiments indicate that the appearance of combining water droplets may be controlled by α intuitively, providing a continuous spectrum of blended surfaces. The other parameter β may be used as another control parameter, for a fixed α, which adjusts the detail of surfaces. However, we find that α is sufficient in this application of water particle smoothing.

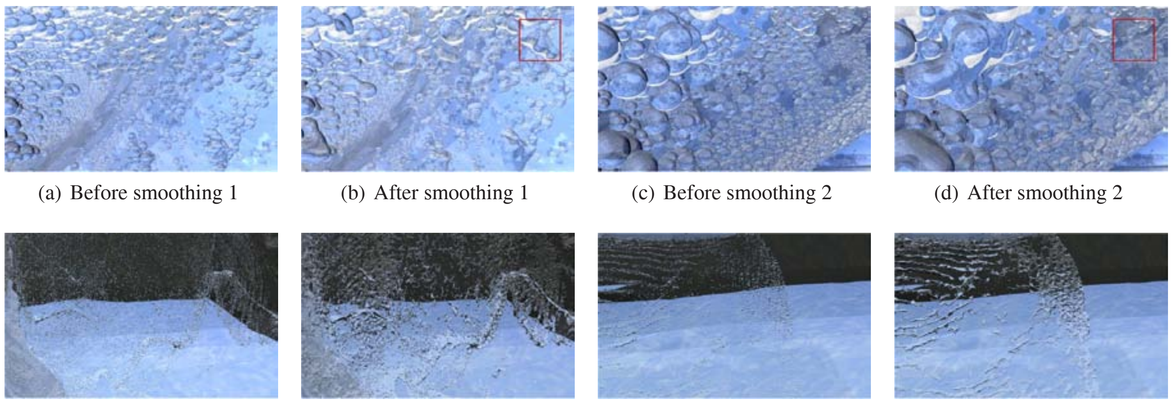
In order to extend the level-set-based ray tracer to handle automatic particle cluster smoothing, we must be able to perform the ray–surface intersection calculation efficiently. In our implementation, a ray is first intersected with the spheres of influence, that is, spheres with radii $\alpha \cdot r_i$ (see Figure 5(e)). Then the ray is marched from the intersection point $\mathbf{x}^{(0)}$ with an initial step size, where a change in the sign of $W(\mathbf{x}) - \beta$ is tested for. This jump-and-check process is repeated with a halved step size until the smoothed surface is crossed. Once that happens, the marching direction is reversed. Ray marching stops when the step size falls below a specified lower bound. If no sign change occurs when the marching is finished, the ray does not intersect the smoothed surface $W(\mathbf{x}) - \beta = 0$. Note that $W(\mathbf{x})$ can be computed by storing the nonuniformly distributed particles in a K-d tree, and examining only a small number of interacting particles.


(a) A particle cluster (b) Blended with $\alpha = 1.5$ (c) Blended with $\alpha = 2.0$ (d) Blended with $\alpha = 2.5$



(e) Ray–surface intersection

Figure 5: *Smoothing of clusters of spheres. The smoothness of blended surfaces is effectively controlled by the kernel's parameter α. (a) to (d) show the gradual changes made by increasing α. (e) illustrates our adaptive ray-marching algorithm for intersecting a ray with smoothed surfaces. By using a K-d tree structure, the intersection computation can be implemented efficiently.*



(a) Before smoothing 1 (b) After smoothing 1 (c) Before smoothing 2 (d) After smoothing 2

(e) Before thickening 1 (f) After thickening 1 (g) Before thickening 2 (h) After thickening 2

Figure 6: *Examples of the smoothing and thickening effects. Figures (a) to (d) illustrate that the smoothing method successfully creates the natural appearance of merging droplets, where (b) and (d) are blended with $\alpha = 2.5$. The rectangle in these figures shows the size of the lowest level cell employed in the level set computation, indicating that the scene is zoomed-in significantly without noticeable aliases. Figures (e) to (h) demonstrate that the thickening technique enables a user to control the appearance of water agglomerates intuitively. Here, the values of d_{intra} and d_{inter} used for (f) and (h) are 0.035 and 0.07, respectively.*

Figures 6(a) to (d) demonstrate the significant differences our smoothing technique makes in rendering close-up images of splashing water. The surface built with the smoothing kernel has a geometric continuity of 2, which is due to the fact that the base spline $P_{[\alpha,\beta]}(r)$ has three multiple roots at $r = \beta$. Notice that the G^2 continuity provides remarkably smoother results in surface blending than does the G^1 continuity.

3.2. Thickening water agglomerates

The appearance of droplet agglomerates resulting from the smoothing process depends on the distribution of water particles computed by the particle system. In our scheme, as explained above, the escaped marker particles are transformed into water particles on a one-to-one basis. Sometimes, there are insufficient marker particles in the underresolved regions, entailing overall deficiencies in the distribution of water particles. A possible way of resolving this is to generate more water particles during the transform, based on a one-to-many mapping for example, but we observe that the particles tend to become scattered quickly during the particle advection unless sufficient numbers of particles are used.

A more effective strategy is to view the set of water particles simulated through the particle system as a skeleton, and to seed more particles between them in the rendering stage. For this, we use two distance measures d_{intra} and d_{inter} to decide where to add new particles. First, at every time step, the simulated water particles are partitioned so that all particles within distance d_{intra} belong to the same group. The idea of the partitioning is to find regions automatically where particles are poorly distributed, and therefore in need of more particles. Once grouping is complete, nearby groups are interconnected by inserting new particles between particles from different groups whose distance is less than d_{inter}. For each selected pair of particles, the number of particles to be seeded is determined from their distance and their radii, so that the new particles with linearly varying radii overlap.

Figures 6(e) to (h) show some examples of the thickening technique in which the additional particles were seeded with $d_{intra} = 0.035$ and $d_{inter} = 0.07$. Our tests with several scenes show that this simple technique enables an animator to control the thickness of water agglomerates intuitively by changing the values of the distance measures appropriately.

3.3. Creating water spray

Water spray is quite useful as a rendering element that enhances the visual realism in water animation. While particles may be exploited to depict the spray effect, the number of particles required can easily become enormous. In work by Takahashi et al. [TFK*03], a volume-rendering technique was applied to render water mist, where mist is sprayed by particles and diffused in the air. We also adopt the volumetric approach, but employ an augmented simulation model to create a natural spray and mist effect.

In our renderer, the water particles are assumed to generate both external force and mist density in the air. For the simulation of spray, we use another Eulerian grid whose resolution may usually be two or four times coarser than that for the level set simulation. At every time step, each particle's velocity is added onto grid points in its neighborhood, weighted by its distance and mass using a kernel function similar to those used in the SPH simulation [DG96,MCG03]. The accumulated velocity $\mathbf{v}_{accum}$ is then added to the external force $\mathbf{f}$ in the incompressible Euler equations:

$$\nabla \cdot \mathbf{u} = 0, \quad \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + (\mathbf{f} + \mathbf{v}_{accum}).$$

simulation time (sec.)			NPT	VMB	VAP
PS	NS	LS			
3.6	15.1	327.0	220,599	1,719.3	0.0065

Table 2: *Statistics on the proposed simulation method. The numbers were averaged per frame for a part of animation sequence from Scene 3 whose average number of processed water particles (NPT) were 220,599. Here, PS, NS and LS indicate the average times taken for updating the particle system, velocity field and level sets, respectively. On the other hand, VMB and VAP respectively denote the volumes of the main body of water and re-entering water particles. The timings were measured on an Intel Xeon 3.6 GHz CPU.*

The updated force drives the velocity field $\mathbf{u}$ in the air, which creates natural flows, reflecting the movement of particles.

$\mathbf{v}_{accum}$ also determines the amount of mist to be added at each time step. In our experiments, we use a simple linear function $S(\mathbf{v}_{accum})$ as a source term. Once generated, the mist with density d is moved along the velocity field using an advection–diffusion type equation:

$$\frac{\partial d}{\partial t} = -\mathbf{u} \cdot \nabla d + \kappa_d \nabla^2 d + S(\mathbf{v}_{accum}) - \alpha_d d,$$

where κ_d is a diffusion constant and α_d is a dissipation rate.

Where previous work [TFK*03] has used a particle's density as a measure of mist generation, we additionally use both velocity and mass. Using the velocity is particularly important, as it creates interesting drag forces as well as inducing more spray in the more turbulent regions.

4. Experimental Results

We have fully implemented the hybrid water simulation technique, and extended our photon-mapping based Monte Carlo ray tracer to generate realistic rendering effects of water droplets and spray effectively. In order to update the Eulerian velocity field, we solved the Navier-Stokes equations based on an unrestricted octree data structure as explained in [LGF04]. In our implementation, the level set computation was performed on a uniform grid using the particle level set method presented in [EFFM02], where a fifth-order accurate Hamilton-Jacobi WENO scheme and a third-order accurate TVD Runge-Kutta scheme were applied to integrate the marker particles and the level set function forward in space and time, respectively, while fast, lower-order discretization methods may enhance the computational efficiency [ELF05]. The escaped water particles were advected using a weighted average of the PIC and FLIP methods as described in subsection 2.3 [ZB05].

Figure 8 gives examples of scenes capturing the turbulent behaviors of splashing water. The extra computational cost incurred when the particle system is integrated into the particle level set method depends on the number of water particles generated during the simulation. However, our experiments with several test scenes indicate that the particle simulation component takes only a small portion of the computation time. For example, when a sequence of frames in the middle of Scene 3, whose average number of processed particles per frame is 220,599, were examined (see Table 2), we observed that only 1.05% of the simulation time was taken by the particle system (3.6 sec.), including the time taken for deciding the radii of newly generated water particles, while the remaining time was spent on the entire particle level set computation (15.1 sec. + 327.0 sec.), with a grid having an effective resolution of $336 \times 144 \times 672$. While we agree that the particle level set part may possibly be implemented more efficiently, it is obvious that the extra cost spent for running the particle system is very small.

The table also reveals that the entire volume of all water particles, re-entering the main body of water, is only 0.0065 per frame, which is minuscule compared to that of the main body (1,719.3), estimated with a smeared Heaviside function. In generating the example animations, we skipped the correction step that updates the level set values since the relative magnitude of the lost volume per frame is just ignorable.

5. Conclusion

We have presented a hybrid water animation technique that focuses on improving the visual realism of splashing water with turbulent motion. We have shown that the minute amount of simulation error that is inevitable with a grid-based method in fact provides a valuable key to the production of complex and fuzzy phenomena, such as tiny water agglomerates and spray. The effectiveness of the presented simulation and rendering methods has been demonstrated using several examples of pictures and animations. Bubbles and foams are another set of important ingredients that greatly improve the illusion of realism for turbulent water. While we have not yet implemented bubbles into our system, they may easily be included by implementing, for instance, the work of Greenwood and House [GH04], which may be considered in parallel with this work.

Acknowledgement

This research was supported by the Ministry of Information and Communication of Korea under the IT Research Center support program.

Appendix: The cubic smoothing kernel $W_i(\mathbf{x})$

Consider a cubic spline function $P_{[\alpha,\beta]}(r)$, made of two cubic polynomials (see Figure 7):

$$P_{[\alpha,\beta]}(r) = \begin{cases} p(r) = ar^3 + br^2 + 1 & 0 \le r \le 1 \\ q(r) = \frac{\beta}{(\alpha-1)^3}(\alpha - r)^3 & 1 < r \le \alpha \\ 0 & r > \alpha \end{cases}$$

The polynomial $q(x)$ has three multiple roots at $r = \alpha$ ($\alpha > 1$), and has value β ($0 < \beta < 1$) at $r = 1$. On the other hand, $p(x)$ has two degrees of freedom after satisfying two constraints $p(0) = 1$ and $p'(0) = 0$. If we additionally require the polynomials to satisfy another two constraints

$p(1) = q(1)$ and $p'(1) = q'(1)$, we obtain a C^1-continuous cubic spline whose unknown coefficients are expressed as: $a = -\frac{3\beta}{\alpha-1} - 2(\beta-1)$ and $b = \frac{3\beta}{\alpha-1} + 3(\beta-1)$.

Now, for a particle i with center $\mathbf{x}_i = (x_i \;\; y_i \;\; z_i)^t$ and radius r_i, define a kernel function $W_i : R^3 \rightarrow R$ as $W_i(\mathbf{x}) = P_{[\alpha,\beta]}(\frac{|\mathbf{x}-\mathbf{x}_i|}{r_i})$. Then the kernel has a compact support domain such that $W_i(\mathbf{x}) = 0$ if and only if $|\mathbf{x}-\mathbf{x}_i| \geq \alpha \cdot r_i$. Also, it has function value β on the particle's spherical boundary, for any value of the radius r_i. It is an important property of our kernel function that is designed to smooth particles with different radii. Furthermore, the property that the base spline has three multiple roots at the boundary of influence enables the kernel to produce a G^2-continuous blended surface. In designing the smoothing function, we have not considered other conditions such as normalization, since the main purpose of our kernel is to find the smoothed boundary of particle clusters.

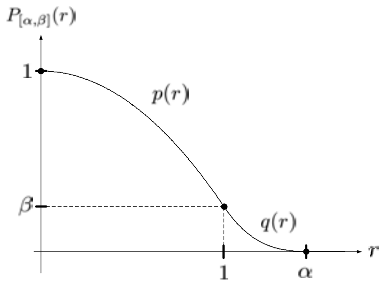


Figure 7: *Base cubic kernel function $P_{[\alpha,\beta]}(r)$. For smoothing spheres of different radii, this base function is scaled in r to account for the radius of each particle. It is made of two cubic polynomials that meet with C^1-continuity at $r = 1$. A quartic polynomial might have been used for $p(r)$ to increase the continuity. However, it is more important to make $q(r)$ have three multiple roots at $r = \alpha$ for visually pleasing smoothing.*

Note that the parameter α controls the general smoothness of blended particle clusters. On the other hand, the second parameter β defines the isovalue for which smoothed isosurfaces are extracted during ray tracing. Because cubic splines may overshoot, it is important to set the value of β properly, as a function of α, so that the resulting spline is monotonic and sigmoidal, as shown in Figure 7. While it might be possible to derive a monotonicity condition that gives a feasible range of β corresponding to a given α, a simple sampling-and-interpolation process is sufficient for our purposes.

Finally, the surface normal of a smoothed surface can be computed using the kernel's gradient vector, which is simply given by $\nabla W_i(\mathbf{x}) = P'_{[\alpha,\beta]}(\frac{|\mathbf{x}-\mathbf{x}_i|}{r_i})\frac{(x-x_i \;\; y-y_i \;\; z-z_i)^t}{r_i|\mathbf{x}-\mathbf{x}_i|}$.

References

[Bli82] Blinn J.: A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 3 (1982), 235–256. 6

[BR86] Brackbill J. U., Ruppel H. M.: FLIP: a method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics 65*, 2 (1986), 314–343. 2, 4

[CBP05] Clavet S., Beaudoin P., Poulin P.: Particle-based viscoelastic fluid simulation. In *Proceedings of ACM SIGGRAPH/Eurographics on Computer Animation 2005* (2005), pp. 219–228. 2

[DG96] Desbrun M., Gascuel M.-P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation 1996* (1996), pp. 61–76. 2, 7

[EFFM02] Enright D., Fedkiw R., Ferziger J., Mitchell I.: A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1 (2002), 83–116. 3, 8

[ELF05] Enright D., Losasso F., Fedkiw R.: A fast and accurate semi-Lagrangian particle levle set method. *Computers and Structures 83* (2005), 479–490. 2, 8

[EMF02] Enright D., Marschner S., Fedkiw R.: Animation and rendering of complex water surfaces. *ACM Transactions on Graphics (ACM SIGGRAPH 2002) 21*, 3 (2002), 736–744. 2

[FF01] Foster N., Fedkiw R.: Practical animation of liquids. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 23–30. 2, 5

[FM96] Foster N., Metaxas D.: Realistic animation of liquids. *Graphical Models and Image Processing 58*, 5 (1996), 471–483. 2

[GH04] Greenwood S. T., House D. H.: Better with bubbles: enhancing the visual realism of simulated fluid. In *Proceedings of ACM SIGGRAPH/Eurographics on Computer Animation 2004* (2004), pp. 287–296. 2, 8

[GM77] Gingold R. A., Monaghan J. J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181*, 2 (1977), 375–389. 2

[GSLF05] Guendelman E., Selle A., Losasso F., Fedkiw R.: Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 973–981. 2

[Har64] Harlow F. H.: The particle-in-cell computing methods for fluid dynamics. *Methods in Computational Physics 3* (1964), 319–343. 2, 4

[HK03] Hong J., Kim C.: Animation of bubbles in liquid. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003), 253–262. 2

[HK05] Hong J., Kim C.: Discontinuous fluids. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 915–920. 2

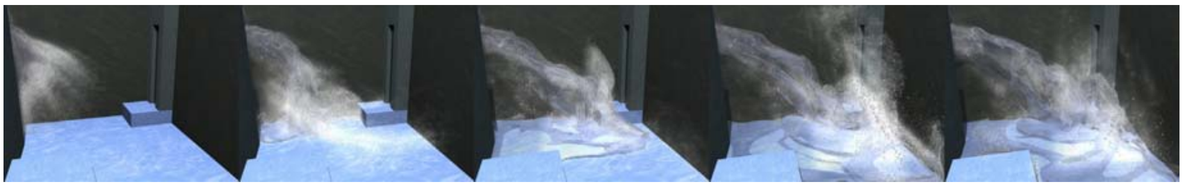
[KTO95] Koshizuka S., Tamako H., Oka Y.: A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Dynamics Journal 4*, 1 (1995), 29–46. 2

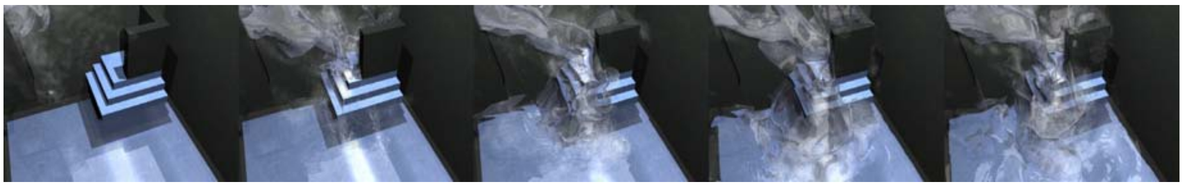
[LGF04] Losasso F., Gibou F., Fedkiw R.: Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (ACM SIGGRAPH 2004) 23*, 3 (2004), 457–462. 1, 2, 8

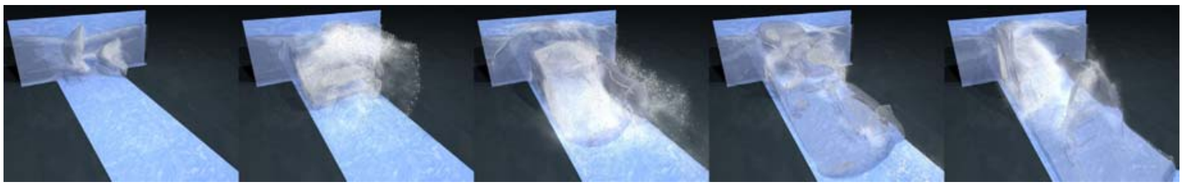
[Luc77] Lucy L. B.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82*, 12 (1977), 1013–1024. 2

[MCG03] Müller M., Charypar D., Gross M.: Particle-based fluid simulation for interactive applications. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003* (2003), pp. 154–372. 2, 5, 7

[MSKG05] Müller M., Solenthaler B., Keiser R., Gross M.: Particle-based fluid-fluid interaction. In *Proceedings of ACM SIGGRAPH/Eurographics on Computer Animation 2005* (2005), pp. 237–244. 2


(a) Scene 1: Pouring water in the room


(b) Scene 2: Pouring water onto the stairs


(c) Scene 3: Flood in the corridor

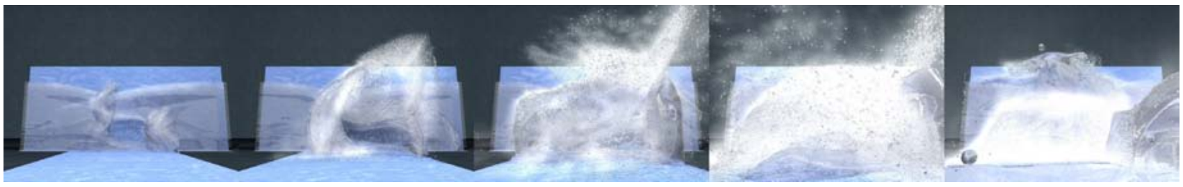

(d) Scene 3: Flood in the corridor - another view

Figure 8: *Animation results. For the three test scenes, the effective grid resolutions employed for the level set computation were* $320 \times 200 \times 560$*,* $320 \times 320 \times 320$ *and* $336 \times 144 \times 672$*, respectively. On the other hand, 109,842, 468,520 and 238,763 particles per frame were processed on average by the particle simulation system to produce the accompanying movies. Again, it should be emphasized that the smoothing and thickening processes are not necessary when the splashes are seen, like these examples, from a distance. These supplementary rendering tools aim to reduce visual aliases that may occur when taking a very close-up view of particle clusters like in the examples of Figure 6.*

[OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *Proc. of Computer Animation 1995* (1995), pp. 198–206. 2

[PTB*03] PREMOŽE S., TASDIZEN T., BIGLER J., LEFOHN A., WHITAKER R. T.: Particle-based simulation of fluids. *Computer Graphics Forum (Eurographics 2003) 22*, 3 (2003), 401–410. 2, 4, 5

[REN*04] RASMUSSEN N., ENRIGHT D., NGUYEN D., MARINO S., SUMMER N., GEIGER W., HOON S., FEDKIW R.: Directable photorealistic liquids. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Computer Animation 2004* (2004), pp. 193–202. 2

[SF95] STAM J., FIUME E.: Depicting fire and other gaseous phenomena using diffusion processes. In *Proc. of ACM SIGGRAPH 1995* (1995), pp. 129–136. 2

[Sim90] SIMS K.: Particle animation and rendering using data parallel computation. In *Proc. of ACM SIGGRAPH 1990* (1990), pp. 405–413. 2, 5

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 910–914. 2

[SSK05] SONG O., SHIN H., KO H.: Stable but nondissipative water. *ACM Transactions on Graphics 24*, 1 (2005), 81–97. 2, 3, 5

[Sta99] STAM J.: Stable fluids. In *Proceedings of ACM SIGGRAPH 1999* (1999), pp. 121–128. 2

[TFK*03] TAKAHASHI T., FUJII H., KUNIMATSU A., HIWADA K., SAITO T., TANAKA K., UEKI H.: Realistic animation of fluid with splash and foam. *Computer Graphics Forum (EUROGRAPHICS 2003) 22*, 3 (2003), 391–400. 2, 7, 8

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (1986), 227–234. 6

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Transactions on Graphics (ACM SIGGRAPH 2005) 24*, 3 (2005), 965–972. 2, 4, 5, 8