Journal of Information Processing Vol.25 486–493 (July 2017)

[DOI: 10.2197/ipsjjip.25.486]

Regular Paper

Particle-based Shallow Water Simulation with Splashes and Breaking Waves

Makoto Fujisawa[1,a)] Takuya Nakada[1] Masahiko Mikawa[1,b)]

Received: September 6, 2016, Accepted: March 3, 2017

Abstract: We propose a fast method of simulating large-scale liquid phenomena by coupling 2D and 3D smoothed particle hydrodynamics (SPH). Our method combines 2D SPH-based shallow water simulation with 3D SPH simulation to effectively treat complex behaviors, such as splashes and breaking waves. To achieve realistic animation, we generate 3D particles from 2D particles by categorizing these particles according to motion and position. One-way interactions between both types of particles are described by the conservation of momentum. We demonstrate the effectiveness of our approach in various graphical scenes.

Keywords: large-scale fluid simulation, shallow water, SPH

1. Introduction

Fluid simulation is widely used to create computer graphics animation of complex natural phenomena. However, it is difficult to compute the details of large phenomena, such as floods and ocean waves, in real-time because of the high computational cost, while interactive applications, such as games, require both high levels of detail and very fast computing. One approach to this type of application is to use a two-dimensional height field, in which three-dimensional fluid behaviors are approximated as surface wave motion. The wave motion is calculated as a two-dimensional flow on a surface, which dramatically reduces the computational cost. However, this method cannot represent some motions characteristic of waves, such as splashes and breaking waves, because it can only treat vertical movement.

In this paper, we propose a new method that realizes fast fluid simulation with finer details by combining two-dimensional shallow-water simulation with particles and three-dimensional smoothed particle hydrodynamics (SPH). A height field represented by the particles is updated according to the shallow water equation (SWE) and three-dimensional particles are generated by analyzing wave movement. Both particles can be naturally combined because the full Lagrangian representation is available. Also, we propose to use a screen-space rendering technique to represent a water surface by using additional particles for rendering in cases where the particle-based shallow-water simulation would cause holes due to an insufficient number of particles.

2. Related Work

For large-scale water simulation, various methods have been proposed. Using a two-dimensional height field instead of a full three-dimensional fluid solver is the most popular solution for interactive applications. Several papers propose using the fast Fourier transform (FFT) to construct a height field and create fluid animations in large-scale scenes with a high level of detail [18], [25], [26]. Kass and Miller [12] introduced a wave equation that can determine water surface movement using a height-field representation. This approach is not based on physics, but it can very effectively simulate large-scale water scenes, such as ocean waves. Later researchers extended this technique to fluid splashing [22], bubbles and droplets [19], waves over terrain [8], [17], and hydraulic erosion [28].

Kass and Miller [12] introduced the shallow water equation (SWE), which is based on the Navier–Stokes equations, for computer graphics animation. This provides a more physically accurate way to simulate the water surface. However they ignored the nonlinear advection term by assuming that water speed is almost constant. Layton et al. [13] used the semi-Lagrangian method to solve the advection term of SWE. Solenthaler et al. [23] extended the method by using particles and SPH. Because they use particles instead of a grid, the method has many benefits, such as ease of extending the simulation space, mass conservation, and simplicity. However, the SWE cannot be used to treat three-dimensional effects, such as splashing and breaking waves. Chentanez et al. [4] combined grid-based shallow-water simulation and the particle-based three-dimensional simulation to represent breaking waves, splashing, and waterfalls. Their method automatically extracts regions of liquid for generating the particles by assuming that such effects arise from rigid interactions. Thürey et al. [27] used triangle mesh patches to simulate breaking waves. Later, Chentanez et al. [5] proposed a method for coupling a grid, particles and a height field to simulate large-scale scenes. A combination of Eulerian and Lagrangian approaches offers the advantages of both methods, although it also suffers some disadvantages of each method. For example, the simulation space is restricted by the grid, but the particle can move freely. Ihmsen et al. [9] proposed a fully Lagrangian method for rendering large-

[1] University of Tsukuba, Tsukuba, Ibaraki 305–8550, Japan
[a)] fujis@slis.tsukuba.ac.jp
[b)] mikawa@slis.tsukuba.ac.jp

© 2017 Information Processing Society of Japan

scale water scenes by generating some types of additional diffuse particles from SPH simulation results. However, this method is not suitable for an interactive application, because it requires a vast amount of diffuse particles in order to generate realistic animations. Our method is also a fully Lagrangian method, but the method based on SPH shallow-water simulation [23], which we extend to three-dimensional effects by using SPH particles. Both the water surface and three-dimensional effects are represented as particles, so that we can easily combine the simulation, and the method has the advantage of being a particle-based method. Also, we propose a method to calculate interactions on the basis of momentum conservation from three-dimensional to two-dimensional particles.

There are many works that consider the full three-dimensional simulation of liquid by using a grid [3] or particles [11]. SPH [6], which is one popular particle-based simulation method, has been widely used for computer graphics [11], [20] because of its simplicity and the ease of extending the simulation space. SPH uses an explicit calculation of pressure according to a state equation, which allows very fast computation but causes incompressibility problems. To solve this problem, various algorithms, such as WCSPH [1], PCISPH [24], IISPH [10], PBF [16], and DFSPH [2] have been developed. We use PBF (Position-Based Fluid) to simulate the behavior of three-dimensional particles because of its handling of incompressibility and its low computational cost relative to other volume-conserving methods.

3. Method

We use SWE with particles [23] to simulate the water wave. This two-dimensional simulation cannot generate any three-dimensional effects, such as breaking waves or splashing. To make such kinds of effects, we introduce particles that move freely in three-dimensional space and then use SPH to calculate the behavior of these particles. The particles are generated via analysis of the movement of SWE particles and deleted when they enter into the water's surface. Finally, both types of particles are rendered by a screen-space rendering technique. In the following sections, we call the particles used in SWE simulation the 2D particles, and those used in three-dimensional PBF are called 3D particles.

3.1 Particle-based Shallow Water Simulation

In general shallow-water simulation, the simulation space is partitioned by a two-dimensional grid, and each grid cell has a height, a vertical velocity, and horizontal velocity, and these variables are updated by using the SWE [13]. SWE, used for representing the motion of the water surface, is based on the Navier–Stokes equations, meaning that the system represents mass and momentum conservation. The velocity field is updated by solving the SWE (2) and then changing the height field on the basis of the velocity Eq. (1).

$$\frac{\partial h}{\partial t} = -\nabla \cdot (h\boldsymbol{u}) \tag{1}$$

$$\frac{D\boldsymbol{u}}{Dt} = -g\nabla(h + H(\boldsymbol{x})) + \boldsymbol{a}_{ext} \tag{2}$$

where h is the height of the water surface, $\boldsymbol{u}$ is the velocity of the water, g is acceleration due to gravity, $H(\boldsymbol{x})$ is the height of the bottom of the water, and a_{ext} is the external force.

In order to realize a seamless interaction between 2D and 3D particles, we use SPH-based shallow-water simulation [23]. SPH-based shallow-water simulation uses the density of SPH to determine the height of the particles.

$$h_i = \frac{\rho_i^{2D}}{\rho_0} \tag{3}$$

where h_i and ρ_i^{2D} are the height from the bottom and the density of particle i, respectively. In this, ρ_0 is the rest density of water. We assume that gravity acts along the y axis. In this case, 2D particles are placed on the xz plane and the density ρ_i^{2D} is calculated as in 2D SPH.

$$\rho_i^{2D} = \sum_{j \in N} m_j^{2D} W_{poly6}(\boldsymbol{x}_j - \boldsymbol{x}_i, l^{2D}) \tag{4}$$

where N is the set of 2D particles within the effective radius l^{2D}, m_j^{2D} is the mass of 2D particle j, and $W(\boldsymbol{x}_j - \boldsymbol{x}_i, l^{2D})$ is the kernel function. By substituting Eq. (3) into Eq. (2), we can obtain the SWE for the particle method:

$$\frac{\partial \boldsymbol{u}_{xz}^{2D}}{\partial t} = -\frac{g}{\rho_0}\nabla\rho_i^{2D} - g\nabla H(\boldsymbol{x}) + \nu\nabla^2\boldsymbol{u}_{xz}^{2D} + \boldsymbol{a}_{ext} \tag{5}$$

where $\boldsymbol{u}_{xz}^{2D}$ is the velocity of a 2D particle on the xz plane and $\nu(= \mu/\rho_0)$ is the kinematic viscosity coefficient. The term $\nu\nabla^2\boldsymbol{u}_{xz}^{2D}$ is a viscosity term used to stabilize the surface motion [14]. Equation (5) updates the velocity $\boldsymbol{u}_{xz}^{2D}$ of 2D particles with timestep length Δt, and position $\boldsymbol{x}_{xz}^{2D}$ is also updated from the velocity. Finally, the y element of the position is calculated from $x_y^{2D} = h_i + H(\boldsymbol{x}_{xz}^{2D})$ and Eq. (3). The SPH shallow-water simulation transforms the change in density to the change in height. We also restrict the velocity by setting a maximum propagation speed of water $\sqrt{gh_i}$ to that given from a wave equation, which creates a more stable simulation.

Boundary handling is a challenging problem of particle-based methods. In our method, we use boundary particles to prevent particle stacking near the boundary, which can be caused by underestimation of the density. Boundary particles are placed into solid boundaries. As an added step, we set the velocity $\boldsymbol{u}_{xz}^{2D}$ to 0 for particles that contact the boundary, which represents a one-way interaction from solid to liquid.

3.2 3D Particle Generation

As mentioned, SWE simulation alone cannot simulate 3D effects, such as breaking waves or splashing. In order to consider these kinds of phenomena, we generate 3D particles from 2D particles. In SWE simulation, the motion of water surface would be restricted to vertical movement and it only generates waves. We assume that there are only two types of 3D effect arising from the wave as shown in **Fig. 1**: a splash shot to the upper direction from crest of wave and a plunging breaker (breaking wave) moved to the horizontal direction from side of wave.

We assume that the phenomena occur when the height of a 2D particle h_i increases rapidly. Thus, the condition for 3D particle generation becomes,

© 2017 Information Processing Society of Japan

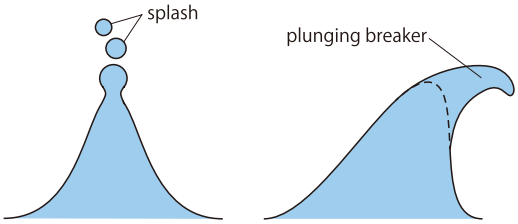


Fig. 1 3D effects caused by a wave.

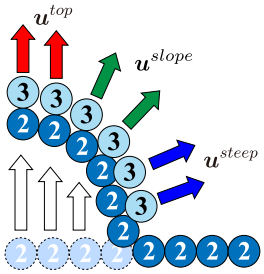


Fig. 2 Classification of newly generated 3D particles and their initial velocities. A blue circle represents a 2D particle and a light blue is a 3D particle.

$$\frac{h_i - h_i^{prev}}{\Delta t} > \gamma_{3D}. \tag{6}$$

Here, h_i^{prev} is the height of the 2D particle i at the previous step, and γ_{3D} is the user-specified threshold for 3D particle generation. Each particle satisfying Eq. (6) is classified into one of three types according to the maximum difference in height with neighboring particles j. This is characterized by

$$h_{diff} = \max(h_j) - \min(h_j) \tag{7}$$

where $\max(h_j)$ and $\min(h_j)$ are the maximum and minimum height among neighboring 2D particles. We classify the 2D particle into the three types, "top", "slope" and "steep" particles, by h_{diff} as shown in **Fig. 2**. The "top" particle represents a splash generated from the top of the wave, while the "steep" particle represents a plunging breaker generated from the steep region of the wave. We also define the "slope" particle in order to smoothly interpolate the motion of the above two particles. These have the following thresholds: top particles are generated when $h_{diff} \le \gamma_{top}$; steep particles are generated when $\gamma_{steep} \le h_{diff}$; and splash particles are generated in other cases (i.e., when $\gamma_{top} < h_{diff} < \gamma_{steep}$), where we assume that $0 \le \gamma_{top} \le \gamma_{steep}$.

The initial position of a new 3D particle is determined as follows.

$$\boldsymbol{x}_{xz}^{3D} = \boldsymbol{x}_{xz}^{2D} \tag{8}$$

$$x_y^{3D} = (1 - \alpha_{height}) h_i^{2D} + \alpha_{height} \max(h_j) + H(\boldsymbol{x}^{2D}) \tag{9}$$

where α_{height} is a parameter used to adjust the height of 3D particle. In this paper, we use 0–0.3 for α_{height}. The initial position is independent of the type of particle. In contrast, the initial velocity of each 3D particle depends on its type.

- "top" particles
  The particle is placed near the wave crest when the change of height from the previous timestep is large but the difference between near particles h_{diff} is small. In this case, splashes are generated, and we set the vertical vector with the velocity:

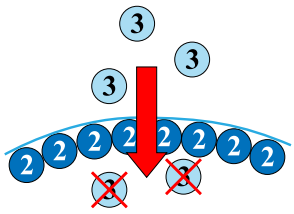


Fig. 3 3D particles deletion.

$$\boldsymbol{u}_{xz}^{top} = 0, \quad u_y^{top} = k_{top} \frac{h_i - h_i^{prev}}{\Delta t}$$

  where k_{top} is the coefficient for "top" particles.
- "steep" particles
  The particle is placed on the side of the wave when the difference from nearby particles is large. In this case, we set an oblique velocity to represent a breaking wave.:

$$\boldsymbol{u}_{xz}^{steep} = \frac{\boldsymbol{a}_{xz}^{2D}}{|\boldsymbol{a}_{xz}^{2D}|} \sqrt{g h_i}, \quad u_y^{steep} = k_{steep} \frac{h_i - h_i^{prev}}{\Delta t}$$

  where k_{steep} is the coefficient for "steep" particles, $\boldsymbol{a}_{xz}^{2D}$ is the vector of acceleration of the 2D particle in the xz plane, and $\sqrt{g h_i}$ is the maximum propagation speed of water as given by a wave equation. We just use a_{xz}^{2D} to determine the direction of the steep particle movement, because the horizontal velocity of the 2D particle does not correspond to the wave speed.
- "slope" particles
  "slope" particles are placed between "top" and "steep," and the initial velocity is calculated by linear interpolation of the corresponding "top" and "steep" particle velocities:

$$\boldsymbol{u}_{xyz}^{slope} = \alpha_{slope} \boldsymbol{u}_{xyz}^{steep} + (1 - \alpha_{slope}) \boldsymbol{u}_{xyz}^{top}$$

  where $\alpha_{slope} = (h_{diff} - \gamma_{top})/(\gamma_{steep} - \gamma_{top})$ is the interpolation coefficient for "steep" particles,

After 3D particles have been generated from the 2D particles, the positions of the 3D particles are updated by PBF [16]. PBF can enforce the incompressibility of the fluid by imposing a density constraint, such as $c_i(\boldsymbol{x}_1^{3D}, \ldots, \boldsymbol{x}_N^{3D}) = \rho_i^{3D}/\rho_0 - 1 = 0$, on all 3D particles. PBF uses a constraint-based method to obtain a uniform distribution of the particles, rather than the Poisson distribution of pressure that is derived from the Navier–Stokes equations. This might produce unreasonable results, because SWE is based on the Navier–Stokes equations. We think the equation for mass conservation (Eq. (1) for SWE) is unlikely to be a problem because both 2D and 3D simulations use the particles for approximation. Also, the pressure p is approximated from the change of height h as $p = g\rho h$ in the SWE [13]. This is similar to a position-based method. In our experiments, we did not observe any visual artifacts related to this problem.

3.3 3D Particle Deletion

We remove 3D particles when their position is under the water surface represented by the 2D particles. If the y coordinate of a 3D particle is less than the height of all neighboring 2D particles (those that lie within the effective radius l^{2D}), then this particle is deleted from the simulation space as shown in **Fig. 3**. Before

© 2017 Information Processing Society of Japan

removing the particle, we update the velocity of the 2D particles within the effective radius l^{2D} of the particle to be deleted, by using following formula:

$$\boldsymbol{u}_{xz}^{2D} = \frac{m_i^{2D}\boldsymbol{u}_{xz}^{2D} + \frac{m_i^{3D}}{N_{neigh}}\left(\boldsymbol{u}_{xz}^{3D} + |u_y^{3D}|\frac{\boldsymbol{r}_{xz}}{|\boldsymbol{r}_{xz}|}\right)}{m_i^{3D}} \tag{10}$$

where N_{neigh} is the number of neighboring 2D particles, $\boldsymbol{r}_{xz}$ is the position of 2D particle relative to the 3D particle on the xz plane. In Eq. (10), the term $m_i^{2D}\boldsymbol{u}_{xz}^{2D}$ represents the momentum of a 2D particle before the velocity update; we add the momentum of the relevant 3D particle to it. For this, we assume that the momentum of a 3D particle $m_i^{3D}u^{3D}$ uniformly affects all neighboring 2D particles. The xz component of the momentum of the 3D particle ($m_i^{3D}\boldsymbol{u}_{xz}^{3D}/N_{neigh}$) is directly added to the momentum of the 2D particles. The y component of the momentum of 3D particle, in contrast, works as a repulsive force on the 2D particles in the direction $\boldsymbol{r}_{xz}/|\boldsymbol{r}_{xz}|$.

3.4 Rendering

General height-field approaches use a flat mesh and move the vertices according to the height field. We cannot apply this method to our simulation results because we incorporate 3D particles. Full 3D particle-based methods, in contrast, commonly use the Marching Cubes algorithm [15] to make the surface mesh from a potential field defined by the particles and the kernel function. This technique is valid for 3D particles only, but we also have 2D particles that are characterized by just upper surface information (i.e., a height). To handle this, we create a surface mesh from both 2D and 3D particles by using the Screen Space Meshes (SSM) algorithm [21] for rendering. SSM uses only those particles visible from a specified viewpoint. Therefore, it is not necessary to calculate information about the particles under the surface wave if the viewpoint is above the 2D particles.

SSM directly projected the particles into screen space and generates a mesh in the 2D space. If the particle distribution is sparse, SSM will generate a mesh with many holes. The particle-based SWE calculates the particle distribution in two dimensional space and then it projects the particles into three dimensional space by computing the height from the density of the particles. Even if there is no hole in two dimensional space, when projecting to three dimensional space with the height, some holes are generated at a region where the density rapidly changes as shown in **Fig. 4**. We solve this problem by using a particle interpolation. The holes appear in the side of the wave where the particles are rapidly moving in a vertical direction. In the side of the wave, the particles also move toward to the center of the wave as indicated by the red arrow in Fig. 4, because the wave is created by the movement of the particles gathering. As a result, the particles placed on the wave side move along to the wave surface. We make use of this feature to fill the holes. **Figure 5** shows the particle interpolation using the particle movement along the wave surface. We simply use a linear interpolation to deside the position of interpolated particles and particle interpolation is performed when the following condition is satisfied.

$$\gamma_{interp} < |\boldsymbol{x}_{xyz}^{2D} - \boldsymbol{x}_{xyz}^{2Dprev}| \tag{11}$$

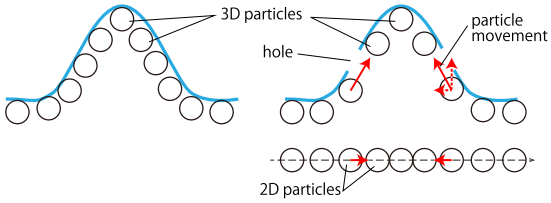


Fig. 4 Holes on surface generated by the vertical movement of the particles. Fully three dimensional simulation generates uniform particle distribution (left), while particle-baesd SWE causes non-uniform distribution in three dimensional space (right).

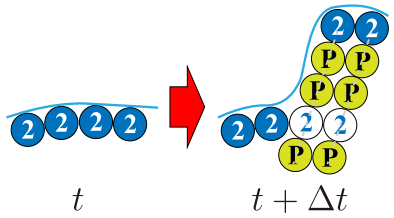


Fig. 5 Particle interpolation for screen-space mesh creation. A green particle represents an interpolated particle.

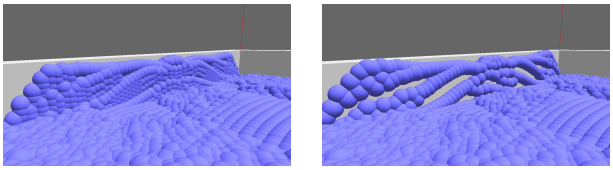


Fig. 6 Comparison of particle interpolation.

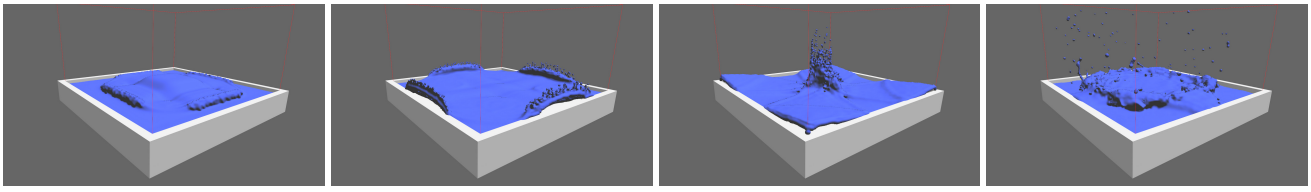
where $\boldsymbol{x}_{xyz}^{2Dprev}$ is the 3D position of a 2D particle at the previous step, and γ_{interp} is the user-defined threshold for particle interpolation. We use $\gamma_{interp} = r^{2D}/2$ for all examples where r^{2D} is the radius of the 2D particle. If the particle fulfills the condition of Eq. (11), then we place new particles on the straight line connecting the previous position to the current position of the 2D particle. In our experiments, the interpolation between the current and previous particles is not enough to cover all holes. So that we extrapolate the particles inside the liquid as shown in Fig. 5 right. **Figure 6** shows the result with and without particle interpolation.

4. Results

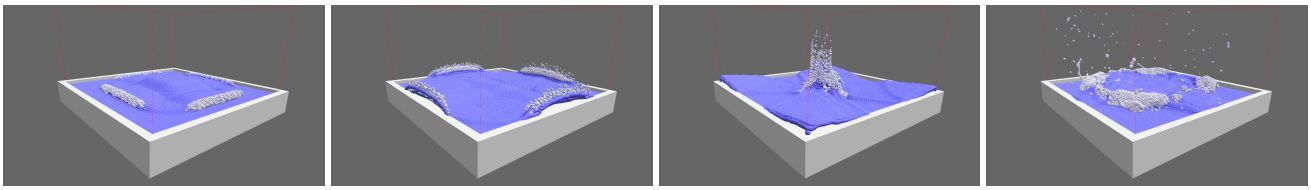
This section describes the results of applying the proposed method to several scenes. All results were generated on a computer equipped with a 3.7 GHz Intel Core i7 CPU and an NVIDIA GeForce GTX TITAN GPU. The algorithm was predominantly implemented on the GPU by using NVIDIA CUDA. We use the method of Ref. [7] to search for neighboring particles. We set the resolution of the screen to 1280×720 pixels and let $\Delta t = 0.005$ s in all examples. **Table 1** lists the parameters used in obtaining the results.

Figure 7 shows a breaking dam scene rendered with both surface mesh and particles. Our method can generate both breaking waves and splashing caused by collisions between several waves. These effects are found in **Fig. 8** made using full 3D simulator with 286,300 particles. We used PBF [16] with 2–10 density iterations per frame and Marching Cubes [15] for surface mesh creation for Fig. 8. These results indicate that the classification we use during 3D particle generation is valid for many scenar-

© 2017 Information Processing Society of Japan



(a) Mesh view



(b) Particle view

Fig. 7 Scene1: Dam breaking from the center region. Blue indicates 2D particles and white indicates 3D particles.

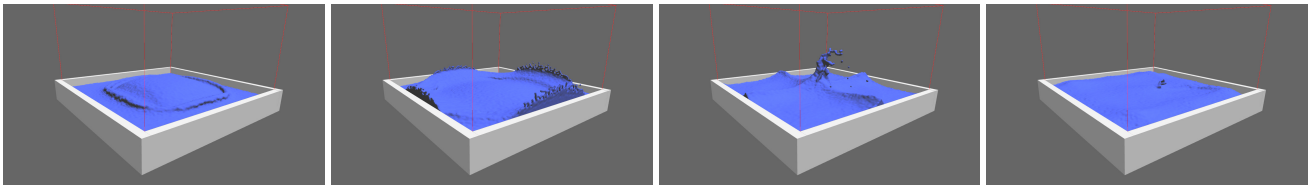
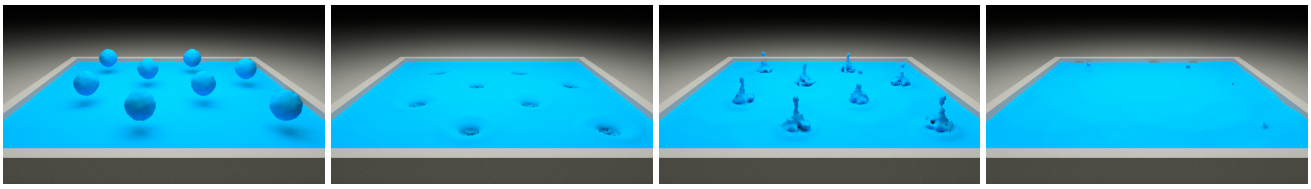
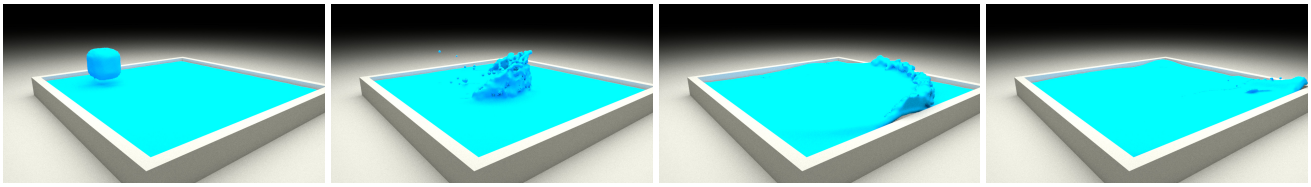


Fig. 8 Dam breaking scene with full 3D simulator. Only 3D particles are used to simulate the fluid behavior. Average simulation time per frame is around 1 fps.



(a) Scene 2



(b) Scene 3

Fig. 9 Water drop falling on a water surface.

Table 1 Paramters for all examples.

	scene 1	scene 2	scene 3	scene 4
α_{height}	0.3	0.3	0	0.3
γ_{3D}	3	1.5	3	1
γ_{top}	0.1	0.15	0.07	0.3
γ_{steep}	0.15	0.2	0.15	0.5
k_{top}	0.3	0.6	0.5	0.35
k_{steep}	0.2	0.2	0.5	0.05

ios. However, we observe that updating velocity while conserving momentum from 3D to 2D particles makes unnaturally high waves, using simulation with 2D particles only as a baseline.

Figure 9 shows that this method of velocity update is valid for a scene with falling water drops. In these scenes, conservation of vertical and horizontal momentums works well to generate waves and splashes. **Figure 10** shows a dam breaking scene with some buildings. Our approach can make plausible animation of large-scale scenes at an interactive rate.

Table 2 summarizes the performance of our method including surface mesh creation. Table 2 also includes the result of full 3D simulation. Similar to the proposed method, we implemented the full 3D simulation on the GPU by using the method of the method of Ref. [7] and we set the 3D particles under the water so that it is the same as Scene 1–4. As shown in Table 2, the proposed method succeeded in drastically reducing the number of the particles. As a result, our method is about 9–23 times faster than the full 3D simulation in average fps and about 5–8.5 times faster in maximum fps. **Figure 11** shows comparisons of the results of Scene 2–4. Parameters for full 3D simulation, such as an effective radius, a mass of particle, are same as the one used for the results of the proposed method.

Figure 12 shows a more detailed comparison of the computational time of Scene 1. In the case of scenes where many particles are stacked in liquid like an ocean, the computation speed of the previous method becomes slower in order to maintain in-

© 2017 Information Processing Society of Japan

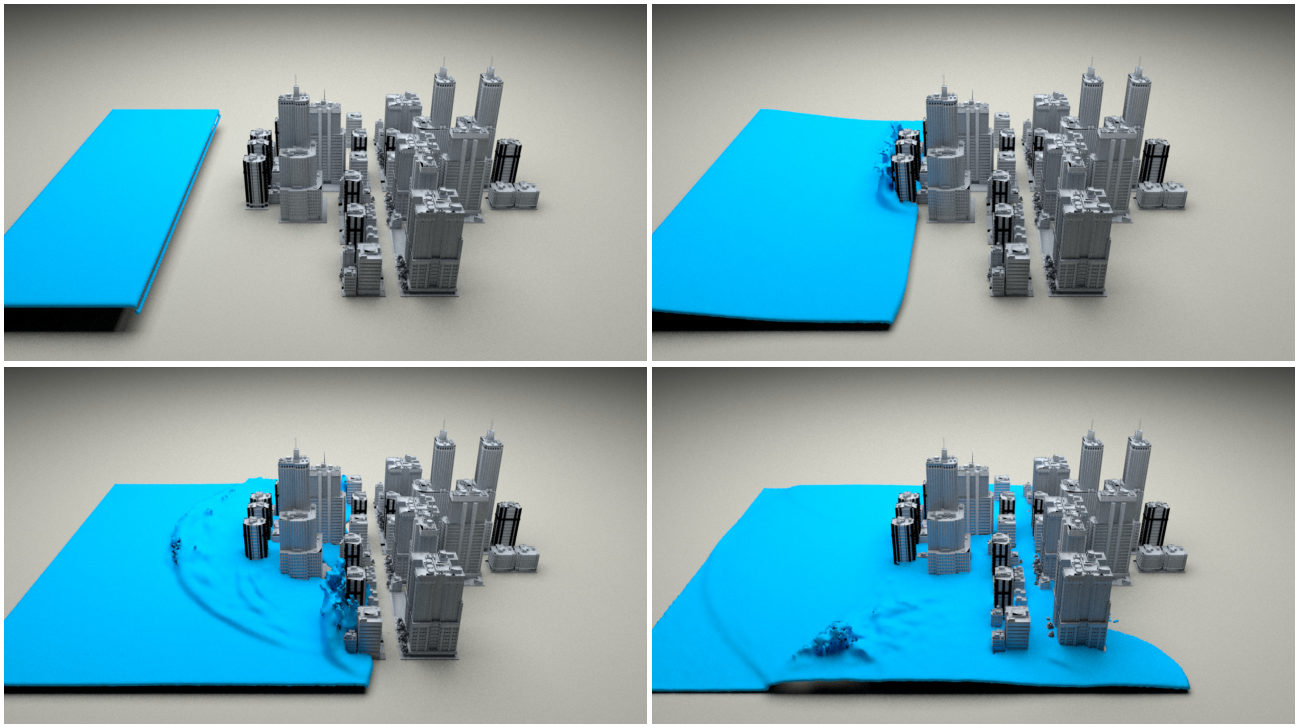
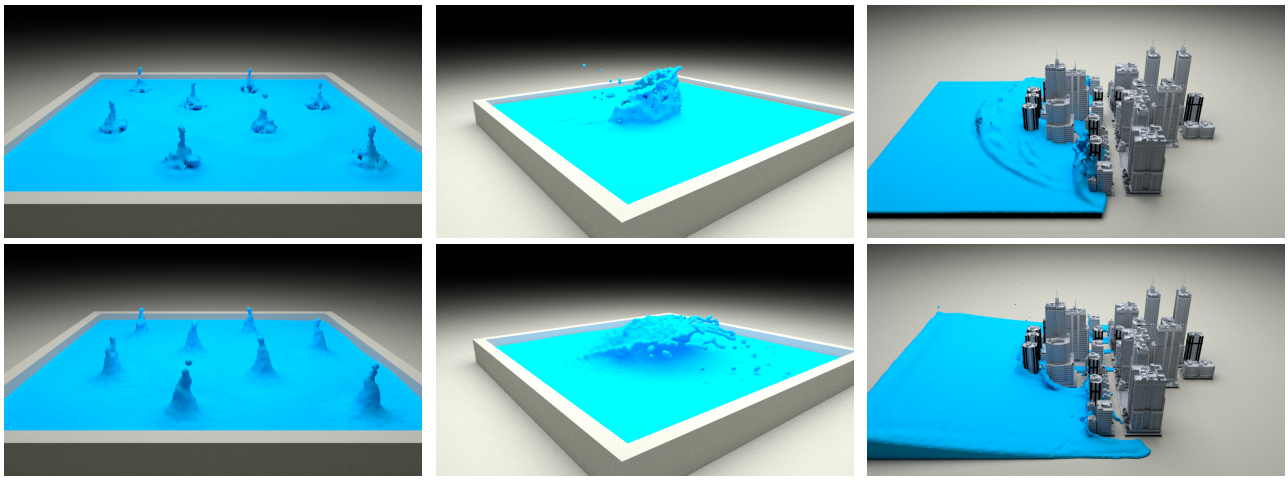


Fig. 10 Scene 4: Dam breaking with buildings.

Table 2 Perfomance results for all examples.

Scene	Proposed method				Full 3D simulation	
	2d partices	Max. 3d particles	Avg. time (fps)	Max. time (fps)	3d particles	Avg. time (fps)
Scene 1	13,700	35,837	61.1	43.5	286,300	6.29
Scene 2	10,800	8,000	137.5	50.0	243,531	5.83
Scene 3	10,800	8,000	98.1	43.5	244,558	5.73
Scene 4	158,775	142,864	24.3	14.3	1,008,315	2.64



Scene 2 Scene 3 Scene 4

Fig. 11 Comparisons of the results of the proposed method (top) and full three dimensional simulation (bottom).

compressibility. On the other hand, the calculation speed of the proposed method does not decrease, since the method only considers the ocean surface. However, the calculation time of the proposed method depends on the number of generated 3D particles as shown in **Fig. 13**. It would cause a temporary decrease in computational speed during simulation.

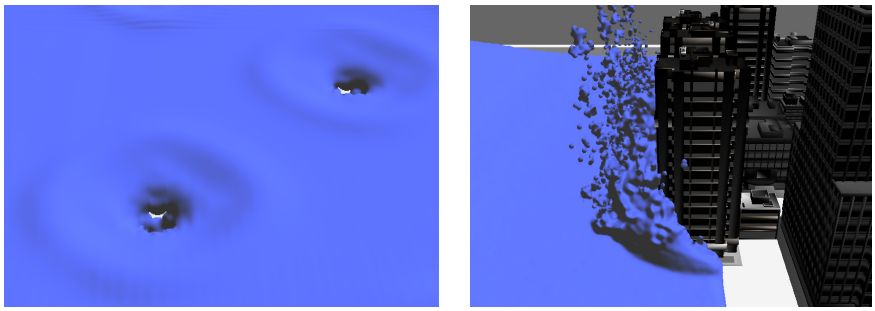
All resulting animations are included in our movie file putting on our website:

http://slis.tsukuba.ac.jp/pbcglab/files/jip_sphswe.mp4

5. Limitation

There are some limitations of the proposed method. At first, the interpolation of particles for rendering cannot fulfill all holes. **Figure 14** (a) shows a concave region generated by a waterdrop falling in Scene 2. The particle interpolation based on the particle movement along the surface cannot treat the region where the

© 2017 Information Processing Society of Japan



(a) (b)

Fig. 14 Limitations of the proposed method: (a) the particle interpolation cannot prevent the hole made in a concave region, (b) inaccurate wave velocity calculation causes unnatural reflected splashes and breaking waves.

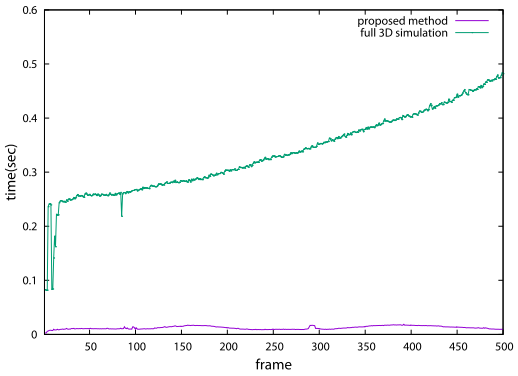


Fig. 12 Comparison of computational time of Scene 1 with full 3D simulation.

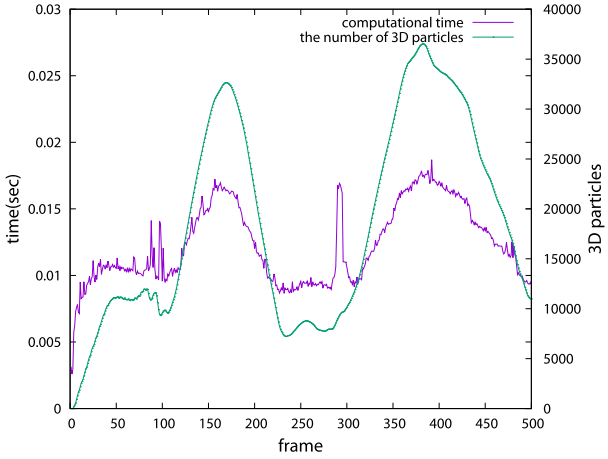


Fig. 13 Computational time of Scene 1 with the number of 3D particles.

particles move outward around a point. Moreover, the shape of the particles clearly appears when the viewpoint approaches and some flickering artifacts are observed at the inner silhouette of the surface mesh due to the SSM algorithm.

Another limitation is that the velocity direction of the wave calculated in Section 3.2 is not entirely correct. Figure 14 (b) shows a frame from Scene 4 with other parameters: $\gamma_{top} = 0.15$, $\gamma_{steep} = 0.2$, $k_{top} = 0.3$, $k_{steep} = 0.3$. The splashes and breaking waves caused from the wave colliding to the building are bounded to the opposite direction of the wave propagation. It is not necessarily the case that the velocity direction of the particles calculated by SWE is equal to the velocity of the wave, because SWE decides a height of wave from the density of the particles.

6. Conclusions and Future Work

We have presented a fully Lagrangian simulation method for large-scale phenomena by combining 2D particle-based shallow-water simulation with a 3D SPH fluid simulation method. 3D particles generated by using a classification based on the state of surrounding 2D particles and a system that updates velocity according to momentum conservation makes it possible to realize various 3D effects, such as breaking waves and splashing.

As future work, we plan to modify the surface mesh creation because the current particle interpolation method is insufficient to fill all holes. We also want to modify flickering artifacts observed in the inner silhouette of the surface mesh. It would be caused by the weak silhouette smoothing in SSM [21], while we could not apply a strong smoothing because the shrinking of the inner silhouette caused another artifact. Moreover, the corrected calculation of the wave direction is also a future work.

We currently consider breaking waves and splashing. There are other possible types of 3D effects, such as waterfalls. Moreover, our method cannot treat a scene like pouring water into a glass because we do not consider mass conservation between 2D and 3D particles. Poured water represented by 3D particles will disappear under the water surface, without increasing the height of the water. We have to consider another height calculation method to achieve mass conservation because the current height calculation is based on only the density of the 2D particles.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 25730069 and 16K00148. We also thank to anonymous reviewers for their constructive comments. The images in this paper were rendered using Mitsuba renderer [*1].

[*1] http://www.mitsuba-renderer.org

References

[1] Becker, M. and Teschner, M.: Weakly Compressible SPH for Free Surface Flows, *Proc. 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.209–217 (2007).

[2] Bender, J. and Koschier, D.: Divergence-free Smoothed Particle Hydrodynamics, *Proc. 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.147–155 (online), DOI: 10.1145/2786784.2786796 (2015).

[3] Bridson, R.: *Fluid Simulation for Computer Graphics*, A K Petters (2008).

[4] Chentanez, N. and Müller, M.: Real-Time Simulation of Large Bodies of Water with Small Scale Details, *Proc. 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.197–206 (2010).

[5] Chentanez, N., Müller, M. and Kim, T.-Y.: Coupling 3D Eulerian,

© 2017 Information Processing Society of Japan

Heightfield and Particle Methods for Interactive Simulation of Large Scale Liquid Phenomena, *Proc. 2014 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.1–10 (2014).
[6] Gingold, R.A. and Monaghan, J.J.: Smoothed Particle Hydrodynamics - Theory and Application to Non-spherical Stars, *Monthly Notices of the Royal Astronomical Society*, Vol.181, pp.375–389 (1977).
[7] Green, S.: CUDA Particles, Technical Report, NVIDIA Whitepaper (2008).
[8] Holmberg, N. and Wünsche, B.C.: Efficient Modeling and Rendering of Turbulent Water Over Natural Terrain, *Proc. 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (*GRAPHITE '04*), pp.15–22 (online), DOI: 10.1145/988834.988837 (2004).
[9] Ihmsen, M., Akinci, N., Akinci, G. and Teschner, M.: Unified spray, foam and air bubbles for particle-based fluids, *The Visual Computer*, Vol.28, No.6-8, pp.669–677 (online), DOI: 10.1007/s00371-012-0697-9 (2012).
[10] Ihmsen, M., Cornelis, J., Solenthaler, B., Horvath, C. and Teschner, M.: Implicit Incompressible SPH, *IEEE Trans. Visualization and Computer Graphics*, Vol.20, No.3, pp.426–435 (online), DOI: 10.1109/TVCG.2013.105 (2014).
[11] Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A. and Teschner, M.: SPH Fluids in Computer Graphics, *Eurographics 2014 - State of the Art Reports*, pp.21–42 (online), DOI: 10.2312/egst.20141034 (2014).
[12] Kass, M. and Miller, G.: Rapid, Stable Fluid Dynamics for Computer Graphics, *ACM SIGGRAPH Computer Graphics*, Vol.24, No.4, pp.49–57 (online), DOI: 10.1145/97880.97884 (1990).
[13] Layton, A.T. and Panne, M.V.D.: A Numerically Efficient and Stable Algorithm for Animating Water Waves, *The Visual Computer*, Vol.18, No.1, pp.41–53 (2002).
[14] Lee, H. and Han, S.: Solving the Shallow Water Equations Using 2D SPH Particles for Interactive Applications, *The Visual Computer*, Vol.26, No.6-8, pp.865–872 (2010).
[15] Lorensen, W.E. and Cline, H.E.: Marching Cubes: A High Resolution 3D Surface Cconstruction Algorithm, *ACM SIGGRAPH Computer Graphics*, Vol.21, No.4, pp.163–169 (1987).
[16] Macklin, M. and Müller, M.: Position Based Fluids, *ACM Trans. Graphics*, Vol.32, No.4, pp.104:1–104:12 (online), DOI: 10.1145/2461912.2461984 (2013).
[17] Maes, M.M., Fujimoto, T. and Chiba, N.: Efficient Animation of Water Flow on Irregular Terrains, *Proc. 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia* (*GRAPHITE '06*), pp.107–115, ACM (online), DOI: 10.1145/1174429.1174447 (2006).
[18] Mastin, G.A., Watterberg, P.A. and Mareda, J.F.: Fourier Synthesis of Ocean Scenes, *IEEE Computer Graphics and Applications*, Vol.7, No.3, pp.16–23 (online), DOI: doi.ieeecomputersociety.org/10.1109/MCG.1987.276961 (1987).
[19] Mould, D. and Yang, Y.-H.: Modeling Water for Computer Graphics, *Computers & Graphics*, Vol.21, No.6, pp.801–814 (online), DOI: 10.1016/S0097-8493(97)00059-9 (1997).
[20] Müller, M., Charypar, D. and Gross, M.: Particle-Based Fluid Simulation for Interactive Applications, *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.154–159 (2003).
[21] Müller, M., Schirm, S. and Duthaler, S.: Screen Space Meshes, *Proc. 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.9–15 (2007).
[22] O'Brien, J.F. and Hodgins, J.K.: Dynamic Simulation of Splashing Fluids, *Proc. Computer Animation 95*, pp.125–132 (1995).
[23] Solenthaler, B., Bucher, P., Chentanez, N., Müller, M. and M.Gross: SPH Based Shallow Water Simulation, *Proc. Virtual Reality Interactions and Physical Simulations* (*VRIPhys*), pp.39–46 (2011).
[24] Solenthaler, B. and Pajarola, R.: Predictive-corrective Incompressible SPH, *ACM Trans. Graphics*, Vol.28, No.3, pp.40:1–40:6 (online), DOI: 10.1145/1531326.1531346 (2009).
[25] Tessendorf, J.: Simulating Ocean Water, *SIGGRAPH 1999 course notes* (1999).
[26] Thon, S., Dischler, J.-M. and Ghazanfarpour, D.: Ocean Waves Synthesis Using a Spectrum-based Turbulence Function, *Proc. Computer Graphics International 2000*, pp.65–72 (online), DOI: 10.1109/CGI.2000.852321 (2000).
[27] Thürey, N., Müller-Fischer, M., Schirm, S. and Gross, M.: Real-time Breaking Waves for Shallow Water Simulations, *Proc. 15th Pacific Conference on Computer Graphics and Applications*, pp.39–46 (online), DOI: 10.1109/PG.2007.54 (2007).
[28] Št'ava, O., Beneš, B., Brisbin, M. and Křivánek, J.: Interactive Terrain Modeling Using Hydraulic Erosion, *Proc. 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.201–210 (2008).



Makoto Fujisawa is currently an Assistant Professor in the Faculty of Library, Information and Media Science, University of Tsukuba since 2011. He received B.Eng., M.Eng., and Ph.D. degrees in mechanical engineering from Shizuoka University in 2003, 2005, and 2008 respectively. He worked for Nara Institute of Science and Technology from 2008 to 2010 as an Assistant Professor. His research interests include computer graphics and physics simulation. He is a member of ACM, IEEE CS, IIEEJ, IPSJ and VRSJ.



Takuya Nakada received B.A. degree from University of Tsukuba in 2015. He is currently working in the NHN hangame Corp. since 2015. His research interests include computer graphics and physics simulation.



Masahiko Mikawa is currently an Associate Professor in the Faculty of Library, Information and Media Science, University of Tsukuba, Japan since 2006. He received B.Eng., M.Eng., and Ph.D. degrees from Osaka University in 1992, 1994 and 2001 respectively. He worked for NTT Access Network Systems Laboratories from 1994 to 2001, NTT Service Integration Laboratories from 2001 to 2003 and was a Lecturer in the Graduate School of Library, Information and Media Studies, University of Tsukuba from 2003 to 2006. He is a member of RSJ, SICE, SOFT and IEEE.

© 2017 Information Processing Society of Japan