芸術科学会論文誌 Vol.11, No.3, pp.92-101

Stochastic Extension Method for Animating Water Flows

Gantulga TSEDENDORJ Norishige CHIBA
Graduate School of Engineering, Iwate University
{ganaa@cg., nchiba@}cis.iwate-u.ac.jp

Abstract
We present a simple and efficient method for animating different types of water flows with emphasis on a surging wave. The method employs a stochastic approach of extending 2D simulation data into 3D animation space. For the reality of a wave motion along the direction of its dominant propagation, we utilize a physically-based, particle simulation at the pre-processing step and extract surface and splash particles only. At the animation step, these 2D surface frames are stochastically sampled along the transverse direction in the animation time and slice (called time-slice) domain with the aid of a user-defined noise function. For the surface reconstruction in 3D, we use a geometrical technique with a smoothing filter in order to remedy undesired 2D artifacts that result from the slice sampling. In conjunction with appropriate supplements, we apply our method to other types of water flows, indicating further interesting applications of a ripple wave and a whirlpool rotation.

Keywords: natural phenomena, physically-based simulation, stochastic extension

1. Introduction

Due to its excessive complexity, animating a large-scale water phenomenon with high performance and computational efficiency still presents a major challenge in the computer animation field. The main reason for this is that a full 3D physically-based simulation (thus animation) requires a tremendous amount of computational work, especially when a large volume of water needs to be simulated. Furthermore, tracking and extracting water surface for rendering is a problem on its own. On the other hand, parametric or spectral approaches based on procedural techniques such as Gerstner wave or Fourier synthesis, are suitable only for animating a relatively calm motion in deep water.

One of the effective ways to solve this problem is through the extension of 2D simulation data into 3D. We start from the reasonable assumption that the transverse motion of the water, is negligible compared to its most dominant motion in the horizontal and vertical directions. This leads to the idea of simulating water wave in 2D and then extending these data into 3D animation. In this setting, the animation space transversally is represented by a sequence of uniformly spaced 2D simulation slices (Figure 1). In this way, it uses a small number of 2D particles in order to create a 3D wave, whereas a direct 3D particle simulation would use all the water particles involved in it. Clearly, by reducing the simulation dimension of the problem, this approach would considerably reduce the computation costs involved in the simulation as well as in the animation steps. However, this dimensional extension approach presents a problem of producing a 3D animation from the limited simulation data, which is an interesting and a complex topic.

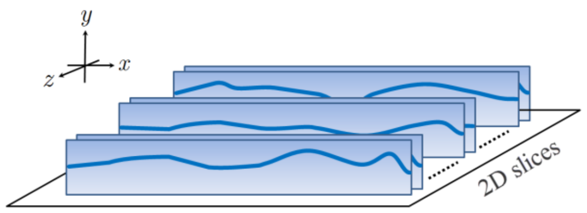

Figure 1. *Animation space is composed of 2D slices (x, y-direction of wave propagation; z-transverse direction)*

In this study, we present a stochastic extension method for animating different types of water flows with emphasis on a surging wave. At the pre-processing step, for generation of 2D wave, we utilize a physically-based, particle simulation, which enables us to capture complicated water dynamics such as wave breaking, merging and collision of two opposite waves with each other. During the wave simulation, we classify water particles and extract only surface and splash particles for the next step. At the animation step, these 2D simulation particles are stochastically sampled in 3D in the animation time and slice (time-slice) domain. Finally, 3D surface is reconstructed by the use of a geometrical technique with a 2D smoothing filter in the direction of wave propagation as well as in its transverse direction.

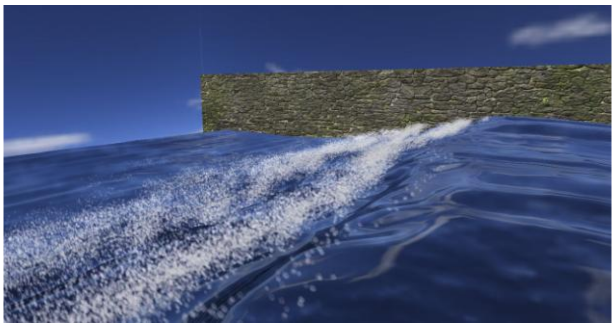

Figure 2. *An example of a surging wave (produced by our method).*

During our simulations, splash-like particles are automatically generated owing to our classification of water particles.

Our method comprises various types of waves, ranging from a ripple wave to a surging wave (except overturning plunging breaker). In view of the fact that a wave is a complicated phenomenon with high degree of complexity, we do not aim to capture all the dynamics of this process, but only to focus on its overall visual realism (Figure 2).

In addition to the surging wave, we apply our method to a ripple wave and a whirlpool, indicating further interesting applications.

We emphasize that any adequate grid or particle-based simulation method can be employed to generate desired water flows in 2D, as discussed above.

2. Related Work

A variety of approaches have been developed to represent water phenomena under various circumstances. Here, we give a brief overview of the most relevant methods.

Early and (still continuing) methods for animating water flows are mostly based on parametric representations for generating procedural water surface in [1] and [2] followed by [3] and [4]. For animation of open water phenomena, such as ocean waves, spectral-type methods have been developed to simulate water surface in [5], [6] and [7]. Since such methods are ultimately based on sinusoidal modeling of the water surface, they are not capable of easily dealing with more complicated water dynamics. A summary of the above procedural methods and their application to modeling and rendering can be found in [8].

With the attempt to obtain more realistic water representation with a splashing effect, [9] and [10] used a height-field model combined with a particle system. The visual quality of water phenomenon was further improved by adding particles for spray and foam in [11]. In [12], full 3D simulation of an overturning breaking wave was performed by controlling the user-defined 2D slice library. A recent work on shallow water simulation for breaking wave can be found in [13], where a real-time plunging breaker with a splashing effect was achieved. This is done by detecting a line along steep wave front and generating a wave patch to represent an overturning effect. Most recently, real-time 3D simulations of various large-scale water scenarios were performed on the GPU in [14] and [15]. The former used a specialized shallow water solver and the latter used Eulerian simulation on a hybrid grid with an optimized multi-grid algorithm.

In order to reduce the overall computational complexity involved in full 3D simulation, researchers have also proposed methods for reducing the number of cells so as to accelerate grid-based fluid simulations by using an octree grid structure in [16] and tetrahedral meshes in [17]. Adaptive sampling algorithms for particle-based simulations are proposed by [18] and [19], focusing computational resources on geometrically complex regions, while reducing the number of particles deep inside the fluid. Furthermore, 3D simulations have been combined with 2D techniques. In [20], a 2D simulation was performed beneath a layer of full 3D simulation, while in [21], a 2D shallow water simulation was coupled to the full 3D free surface fluid simulation.

In addition to these approaches, dimensional extension approach of synthesizing 2D simulation into 3D animation was proposed by [22] for animating a wave by synthesizing 2D velocity field, whereas in [23] a large-scale explosion was animated by using cylindrical interpolation with Kolmogorov spectrum. In [24] and [25], a noise-based synthesis was used in order to animate a surging breaker. Lately, dimensional extension was used to simulate a highly-detailed fire on parallel GPUs in [26].

In this study, we extend our method in [27] which is a development of the idea proposed by [24] with the following contributions:

- Introduction a new stochastic term which provides our method more controllability;
- Detailed investigation on the usage of the *fractional Brownian motion* (fBm) noise in particular, effect of *Low/High-Pass Filter* (LPF/HPF) and the parameter β (relation to complexity/shape of wave front);
- Polygon generation for rendering, including surface particle detection, LPF for slice sampling;
- Application to other types of water flows such as a ripple wave and a whirlpool rotation.

Our animation method consists of the following steps: first, we generate two different types of water waves in 2D and classify water particles in the MPS method (Section 3). Next, in Section 4, these surface wave slices are sampled in the time-slice domain by our stochastic sampling method in 3D. The final wave surface is reconstructed by use of a geometrical technique with the Gaussian filter. Finally, rendering is performed with the aid of environmental mapping taking into account the primary optical properties of water such as reflection, refraction and Fresnel effect with the surrounding illumination (Section 5). The paper is concluded with the discussion and remarks on future work (Section 6).

3. Simulations in 2D

In this section, we classify water particles by a density-based approach and generate desired wave motions using the MPS method (for details see [28], [29]).

3.1 Simulation Method

Generally, water dynamics can be described by the following differential equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{u} = 0 \qquad (1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \qquad (2)$$

where **u** is velocity, *t* is time, *ρ* is density, *p* is pressure, *ν* is viscosity, **f** is external force.

The MPS method is designed to solve Equations (1) and (2) via particle interactions. After applying the pressure projection scheme to Equation (2), the MPS method discretizes and transforms them into particle interaction equations. All interactions between particles are limited to within a given cut-off radius r_e. The weight of interaction between two particles is defined as

$$w(r) = \begin{cases} r_e / r - 1, & 0 \le r < r_e \\ 0, & r \ge r_e \end{cases}$$

where *r* is the distance between two particles *i* and *j* at positions r_i and r_j respectively,

$$r = |\mathbf{r}_j - \mathbf{r}_i|$$

In the MPS method, local density of the fluid is represented by *Particle Number Density* (PND). PND for particle *i* is defined as

$$n_i = \sum_{j \neq i} w(|\mathbf{r}_j - \mathbf{r}_i|)$$

Because no particles exist in the outer region of a free surface, the PND decreases for particles that are on or over the free surface. Thus, a water particle *i* that satisfies the condition:

$$n_i < \delta n^0 \qquad (3)$$

where n^0 is the standard PND and *δ* is a threshold, is considered to be on the free surface. Finally, solid boundaries such as a wall or other stationary objects are represented by fixed particles (with zero velocity).

3.2 Classification of Water Particles

In order to cope with surface reconstruction in 3D, we first detect surface particles during the simulation using a simple density-based approach. It is worth noting that any adequate surface tracking method (for overview see [30]) can be used in our case, since 2D simulation as well as the corresponding surface particle extraction is performed at the pre-processing step.

We classify water particles depending on their PNDs. Advantage of this simple classification is that it gives us a reasonable approximation of surface and splash-like particles with no special treatment or computation. A water particle is classified according to Definition (3) with the following modification: particle *i* is considered to be

$$\begin{aligned} &\textit{a splash particle, if} \quad n_i < \delta_1 n^0 \\ &\textit{on the surface, if} \quad \delta_1 n^0 \le n_i \le \delta_2 n^0 \\ &\textit{in the deep water, if} \quad n_i > \delta_2 n^0 \end{aligned} \qquad (4)$$

where δ_1 and δ_2 are user-defined thresholds, is satisfied.

3.3 Wave Simulations

In our method, simulations of two different waves are carried out. The basic routine of our simulation by the 2D MPS method is summarized in Algorithm 1.

Algorithm 1. *Simulation of Water Wave*

```
1.  Initialize simulation environment
2.  for each time step
3.      Move pushing wall (in case of surging wave)
        or Add particles (in case of ripple wave)
4.      Compute and apply forces to particles
5.      Advect particles
6.      Set neighborhood and calculate PND
7.      Extract surface/splash particles by (4)
8.      Delete particles (in case of ripple wave)
9.  end for each
```

Some important simulation parameters and their values that used in our experiments are shown in Table 1. The simulation container of each case is represented by fixed particles, while water is represented by regular particles. In our experiments, two main types of particles are used in each simulation (Table 2).

Table 1. Simulation parameters & values used

Parameter	*Notation*	*Value*
Density	ρ	1000
Viscosity	ν	0.0011
Gravity	$\mathbf{g}$	9.81
Boundary parameters	δ_1 / δ_2	0.49 / 0.96
Timestep	Δt	0.001

As noted previously, the classification of water particles drastically reduces (up to 26 times!) the total number of particles involved in the animation step (Table 3).

Table 2. Num. of particles *during* simulations

Simulation	***Water particles***	***Fixed particles***	***Time /hrs./ (iterations)***
Surging wave	8376	1662	3.74 (12K)
Ripple wave	6884	1134	2.11 (10K)

The variation of the number of particles are associated with the varying numbers of water body, surface and/or splash particles at each simulation time step in each particular simulation.

Table 3. Approx. num. of particles *after* simulations

Simulation	***Surface/splash particles***	***Particle reduction /times/***
Surging wave	322	26
Ripple wave	284	18

In the case of the whirlpool, the same simulation data as for the ripple wave is used.

3.3.1 Surging Wave

For generation of a surging wave (with no overturning), we use a piston-type method with a pushing wall. The pushing wall periodically moves back and forth in a random manner, maintaining wave motion during the entire simulation (Figure 3*a*). The corresponding screenshot of the simulation is shown in Figure 4*a*.

3.3.2 Ripple Wave

In the case of a ripple, we employ a similar way to that described in [31] by using an *inflow-outflow scheme* together with the *particle recycling strategy* (Figure 3*b*). The screenshot is shown in Figure 4*b*.

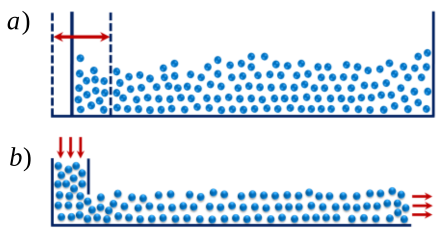

Figure 3. *Simulation environments for a) surging and b) ripple waves.*

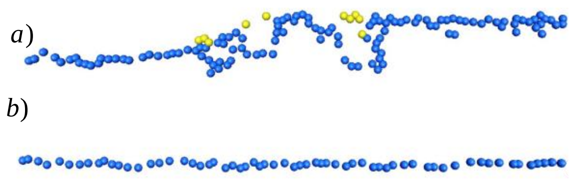

Figure 3. *Screenshot of a) surging and b) ripple waves. Blue-surface particles, yellow-splashes.*

4. Animations in 3D

In this section, we proceed in extending 2D surface waves obtained from the simulation step, into 3D by the use of our stochastic extension method. For the 3D surface reconstruction, we use a 2D Gaussian filter.

4.1 Stochastic Extension Method

First of all, it is easy to imagine that if a single 2D wave surface were to be duplicated in 3D, the resulting surface would look absolutely uniform along the transverse direction (Figure 8*a*). Our method removes this unnatural uniformity of sampling the same 2D simulation frame at every animation time step.

Our slice sampling process is performed by the following stochastic formula:

$$s_k^i = [(i + F \cdot N(i,k)) \cdot A] \qquad (5)$$

$$F \cdot N(i,k) \le 1 \qquad (6)$$

where s^i_k is simulation frame, $i=0,1,...,T-1$, is animation time, $k=0,1,...,K-1$, is animation slice, N is noise function, F is parameter for noise fluctuation range and parameter A controls the animation speed. Notation [.] is for the nearest integer.

In order to describe how our method works, let us consider a simulation frame sequence at animation time interval [i, $i+1$]. Simulation frame s^i_k for current slice k is selected according to the stochastic Formula (5) under Constraint (6) as shown in Figure 5. Once all the slices are filled out by randomly selected frames, the slices are linearly sampled for the construction of a 3D wave shape. Constraint (6) ensures that no abrupt backward and/or forward frame transitions occur during the course of entire animation.

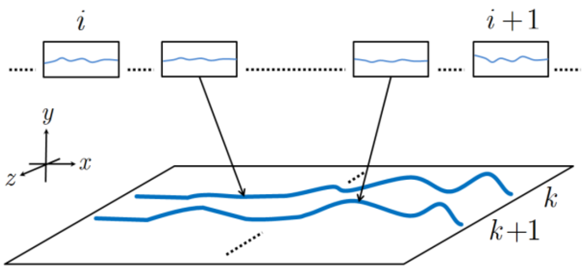

Figure 5. *Slice sampling process.*

For the replacement of the noise function N, we take *fractional Brownian motion* (fBm or also known as $1/f^{\beta}$) noise. The fBm noise is fully controllable by its power spectrum which is inversely proportional to frequency f to the power of the parameter β. The value of β determines the noise correlation (Figure 6). The larger value of β, the smoother noise and zero corresponds to a white noise. It is well known that the fBm noise is observed in many natural phenomena, such as fluid phenomena that exhibit a *waving* pattern [32]. In our method, we have implemented 2D fBm noise through spectrum synthesis technique based on the Fast Fourier Transform [33].

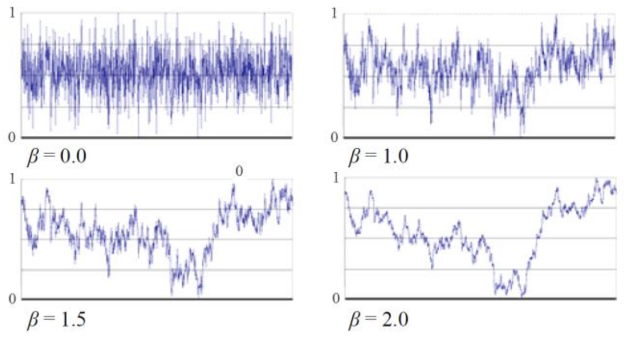

Figure 6. *Examples of 1D fBm noise with various β.*

4.2 Surface Approximation

In order to deal with surface reconstruction, we use a height-field technique with the Gaussian filter. This allows us to reconstruct water surface in 3D with a reasonable approximation, while remedying line-like artifacts that result from the sampling of 2D slices.

To start with the height-field construction, we create a uniform base mesh of size $N \times M$ on the XZ horizontal plane as follows:

$$I_{XZ} = \{(x_n, z_m) : x_n = n\Delta x, z_m = m\Delta z\}$$

where $n=0,..,N\text{-}1, m=0,..,M\text{-}1$; $\Delta x, \Delta z$ - mesh steps.

As soon as the slice sampling is done by Formula (5) and (6), height for the center point $P=(P_x, P_z)$ of each cell $[x_n, x_{n+1}] \times [z_{m-1}, z_{m+1}]$ of the base mesh I_{XZ} is approximated by the following weighted average:

$$h(P) = \sum_l G_l(x,z) h_l \Big/ \sum_l G_l(x,z) \qquad (7)$$

where $G(x,z) = e^{-((x-P_x)^2 + (z-P_z)^2)/2\sigma^2}$ is the Gaussian weight and h_l's are surface particles' heights in the neighboring cells.

For a better result, we insert an extra slice between two neighbor slices and set cut-off area of size $[3\Delta x, 2\Delta z)$ for the Gaussian filtering with $\sigma = 2.0$ (Figure 7).

Throughout the surface reconstruction process, the corresponding splash particles are left untouched.

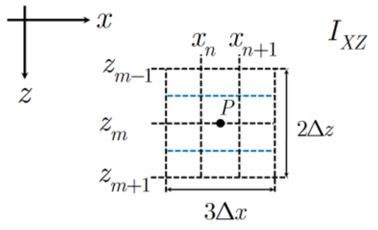

Figure 7 (Top view) *Cell center P on base mesh I_{XZ} (blue dashed lines represent inserted slices).*

4.3 Surging Wave

In order to obtain a 3D wave from 2D slices, we have assumed that the transverse motion of the wave is negligible compared to its motion in the horizontal and vertical directions. This assumption allows us to perform a straightforward sampling of the 2D surface waves along the transverse direction in the time-slice domain in 3D. An outline of our animation process is shown in Algorithm 2*a*.

As mentioned before, one of the advantages of our method is that it has more flexibility that provides more parameter controllability. Regenerating the noise N and/or controlling its fluctuation F leads to a variety of surface shapes: varying from a unrealistic looking surface (Figure 8) to a more realistic one (Figure 9).

Algorithm 2*a*. Animation of Surging Wave

```
1.  Initialize 2D slices
2.  for each time step
3.      for each slice
4.          Set current slice by (5) and (6)
5.      end for each
6.      Reconstruct surface by (7)
7.  end for each
```

By observing waves in nature, it can be seen that a wave front has rather a smoother shape, not a sharp, rough shape with abrupt changes. To achieve this effect, we generate low-pass filtered fBm noise.

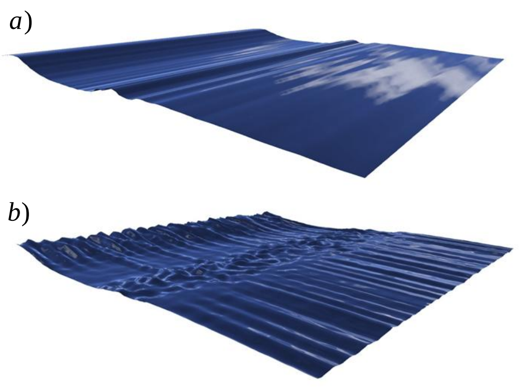

Figure 8. *Surging wave sampled with a) plain duplication i.e., F=0 and b) HPF, F=0.5; β=1.5.*

Moreover, the use of the LPF and a larger value of the parameter β avoids to create undesired zigzag-like connectivity between neighboring slices (see Figures 8 and 9 for comparison).

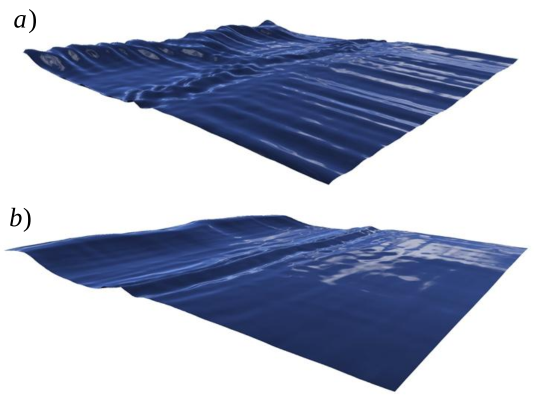

Figure 9. *Surging wave sampled with LPF and F=0.5 a) β=1.5 and b) β=2.8.*

4.4 Ripple Wave

In the case of a ripple wave, we proceed in a slightly different way by synthesizing several waves in different directions. In this way, it eliminates the uniform appearance along the transverse direction which would arise from the motion in a single direction.

To begin this process, we consecutively apply the slice sampling and surface reconstruction processes to each particular direction. For the approximation of the final height *h(P)* for each cell center *P* on the base mesh I_{XZ}, we synthesize heights of the closest neighbors in each direction by

$$h(P) = \sum_d \sum_Q \omega(h_d(Q)) \qquad (8)$$

where ω is user-defined weight and h_d are heights of neighbor points *Q*'s in direction *d* (Figure 10).

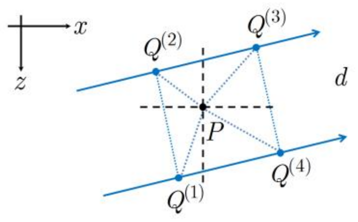

Figure 10 (Top view) *Height approximation in direction d.*

Our synthesis algorithm for a ripple wave is outlined in Algorithm 2*b*. In our experiment, we approximate final heights as an average of bi-linearly interpolated neighboring heights in two different directions. It is experimentally found that about 30º difference between these directions produces a better result.

Algorithm 2*b*. Animation of Ripple Wave

```
1.  Initialize 2D slices in each direction
2.  for each time step
3.      for each direction
4.          for each slice
5.              Set current slice by (5) and (6)
6.          end for each
7.          Approximate surface by (7)
8.      end for each
9.      Synthesize heights by (8)
10. end for each
```

4.5 Whirlpool

Whirlpool is another example to show the flexibility of our method. For whirlpool animation, after applying Formula (5) and (6) to the slices, following Algorithm 2*a*, we transform them into spirals:

$$x(t) = ae^{bt}\cos(t); \;\; z(t) = ae^{bt}\sin(t)$$

where t is parameter in $[0,2\pi]$ and a, b are constants. For slice sampling, we use a radial sampling so that the spiral ends are centered round the whirlpool center. In this way, it provides more turbulence closer to the center (Figure 11).

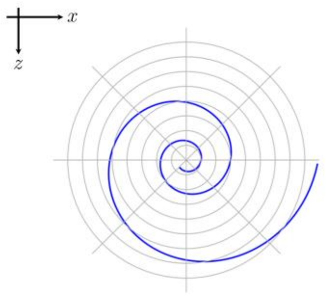

Figure 11 (Top view) *Blue curve represents single spiral-shaped slice.*

In our experiment, we take a=15Δx and b=25Δz.

5. Results

We use environmental cube mapping for rendering each case of the water phenomena, taking into account the primary optical properties of water such as reflection, refraction and the Fresnel effect with the surrounding illumination. In the case of the surging wave, we generate additional artificial splash particles in spherical region around the actual ones. These additional splashes have random properties such as initial positions, velocities, and lifetimes, based on those of the actual splash particles and are rendered by point sprites.

For performance evaluation purpose, we implemented each animation on the CPU and used a machine Intel(R) Core(TM) Q9550 2.83GHz with 4GB RAM. The implementation of each animation is simple, straightforward and the code is written in C++ using OpenGL libraries (with no optimization). The size of the (normalized) fBm noise is $T=16 \times K=256$ in the time-slice domain, except the case of whirlpool, where only $K=128$ slices are used. An example of the 2D fBm noise that we used in our experiment of surging wave animation is shown in Figure 12. The values used for the noise parameters are provided in Table 4, where c is cut-off frequency of the LPF in each animation. Animation parameters and results are shown in Table 5. Parameter A which is inversely proportional to the simulation time step Δt=0.001, is taken to be $4/\Delta t$.

Table 4. Noise parameters

Animation	*c*	*β*	*F*
Surging Wave	128	2.0	0.3
Ripple Wave	128	2.5	0.4
Whirlpool	64	2.5	0.4

Table 5. Animation parameters

Animation	*Resolution* (*N×M*)	*Slices used* (*K*)	*fps*
Surging Wave	128×128	256	12
Ripple Wave	128×128	256	10
Whirlpool	128×128	128	17

We provide screenshot sequences of our animations in Figures 2, 13 and 14.

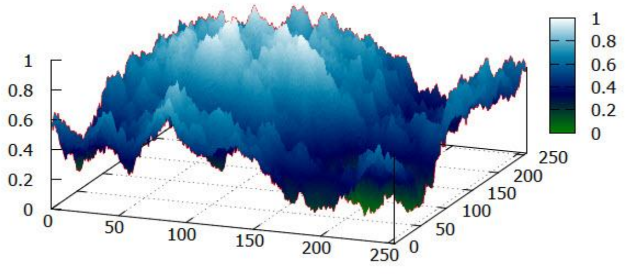

Figure 12 The *2D fBm noise used in our animation, where β=2.*

6. Conclusions and Future Work

In this study, we present an efficient method for creating a 3D animation from the limited 2D simulation data. It uses a small number of 2D surface particles in order to create a 3D wave, whereas a direct 3D particle simulation would use all the water particles involved in it. For instance, a full 3D MPS simulation would require over half a million particles to animate a surging wave of our scale (see Tables 2 and 3 for comparison).

Our stochastic extension method combined with 2D physically-based, particle simulations is used to animate different types of water flows, ranging from a ripple to a surging wave, including a whirlpool-like rotation. It also seems feasible to apply the presented method to other types of water flows, for instance, animating a fountain with cylindrical slice sampling. It is simple to implement our animation method as it requires no knowledge of advanced physical modeling (except simulation part).

The main limitation in the method is that the water must have the dominant flow direction, since the final 3D animation is composed of 2D slices. This restriction can be dropped in some cases as we did it in the ripple and whirlpool animations. Furthermore, to allow the re-use of 2D simulation frames, we assume that the terrain under water body should be uniform in the transverse direction. Our animations do not allow interactions with external objects and it should be further investigated. Our method can be used to produce background or outdoor scenes in applications where interactivity is not required.

As the main part of future work, we would like to focus on in extending our method so that the animations could run for a longer time period than the pre-computed 2D simulation data would allow. For comparison purpose, we would also like to implement our animations on the GPU.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 24300032.

References

[1] A. Fournier and T. Reeves. A simple model of ocean waves. In *Proceedings of SIGGRAPH*, Vol. 20, pp.75-84, 1986.

[2] D. R. Peachey. Modeling waves and surf. In *Proceedings of SIGGRAPH*, pp.65-74, 1986.

[3] J. C. Gonzato and B. L. Saec. On modeling and rendering ocean scenes, *Journal of Visualization and Computer Simulation*, Vol. 11, pp.27-37, 2000.

[4] S. Jeschke, H. Birkholz, and H. Schmann. A Procedural Model for Interactive Animation of Breaking Ocean Waves, *WSCG'2003 POSTERS Proceedings*, 2003.

[5] G. Mastin, P. Watterberg and J. Mareda, Fourier synthesis of ocean scenes, *IEEE Computer Graphics and Applications*, Vol.7, No.3, pp.16-23, 1987

[6] S. Thon, J. M. Dischler and D. Ghazanfarpour. Ocean waves synthesis using a spectrum-based turbulence function, In *Proceedings of the International Conference on Computer Graphics*, IEEE Computer Society, pp.65, 2000

[7] J. Tessendorf. Simulating Ocean Surfaces. *SIGGRAPH Course Notes*, 1999 (updated in 2001 and 2004).

[8] E. Darles, B. Crespin and D. Gazanfarpour. A survey of ocean simulation and rendering techniques in computer graphics, *Computer Graphics Forum*, Vol. 30, No. 1 pp.43-60, 2011.

[9] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Proceedings of the Computer Animation*, IEEE Computer Society, pp.198, 1995

[10] M.M. Maes, T. Fujimoto and N. Chiba. Efficient animation of water flow on irregular terrains. In *Proceedings of Int. Conference on Computer graphics and Interactive techniques*, GRAPHITE'06, pp.107-115, 2006.

[11] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka and H. Ueki. Realistic animation of fluid with splash and foam. *Computer Graphics Forum*, Vol. 22, No. 3, pp.391-400, 2003

[12] V. Mihalef, D. Metaxas and M. Sussman. Animation and Control of Breaking Waves, In *Proceedings of the SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.315-324, 2004.

[13] N. Thurey, M. Muller-Fischer, S. Schirm and M. Gross. Real-time breaking waves for shallow water simulations. In *Proceedings of the Pacific Conference on Computer Graphics and Applications*, pp.39-46, 2007.

[14] N. Chentanez and M. Muller. Real-time simulation of large bodies of water with small scale details. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.197-206, 2010.

[15] N. Chentanez and M. Muller. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pp.1-10, 2010.

[16] F. Losasso, F. Gibou and R. Fedkiw. Simulating water and smoke with an octree data structure. In *Proceedings of the ACM SIGGRAPH*, pp.457-462, 2004.

[17] N. Chentanez, B. E. Feldman, F. Labelle, J. O'Brien and J. Sewchuk. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the ACM SIGGRAPH/ Eurographics Symposium on Computer Animation*, pp. 219-228, 2007.

[18] B. Adams, M. Pauly, R. Keiser and L. Guibas. Adaptively sampled particle fluids. ACM *Transactions on Graphics*, Vol. 26, No. 3, pp.48, 2007.

[19] W. Hong, D. House and J. Keyser. Adaptive particles for incompressible fluid simulation. *The Visual Computer*, Vol. 24, No. 7, pp.535-543, 2008.

[20] G. Irving, E. Guendelman, F. Losasso and R. Fedkiw. Efficient simulation of large bodies of water by coupling two and three Dimensional Techniques. *ACM Transactions on Graphics*, Vol. 25, pp.805-811, 2006.

[21] N. Thurey, U. Rude and M. Stamminger. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.157-164, 2006.

[22] S. Thon and D. Ghazanfarpour. Real-time Animation of Running Waters Based on Spectral Analysis of Navier-Stokes equations. In *Proc. of Computer Graphics International*, pp.333-346, 2002.

[23] N. Rasmussen, D. Q. Nguyen, W. Geiger and R. Fedkiw. Smoke simulation for large scale phenomena. *ACM Trans. Graph.* Vol. 22, No. 3, pp.703-707, 2003.

[24] T. Fujimoto, S. Miyauchi, T. Suzuki and N. Chiba. Noise-based Animation of Waving Phenomena. In *Proc. IWAIT2005, IEICE technical report. Image engineering.* Vol. 104, No.545, pp.93-98, 2005.

[25] Q. Wang, Y. Zheng, C. Chun, T. Suzuki, T. Fujimoto and N. Chiba. Efficient Rendering of Breaking Waves Using MPS Method. *Journal of Zhejiang University SCIENCE A.* Vol.7, No.6, pp.1018-1025, 2006.

[26] C. Horvath and W. Geiger. Directable, high-resolution simulation of fire on the GPU. *ACM Trans. Graph.* Vol. 28, No.3, pp.1-8, 2009.

[27] G. Tsedendorj and N. Chiba. An efficient method for animating breaking wave. *NICOGRAPH International conference*, Fluid Simulation section, No. 3-2, 2011.

[28] S. Koshizuka, A. Nobe and Y. Oka. Numerical Analysis of breaking waves using the MPS method. Int. *J. Numer. Methods in Fluids*, Vol. 26, pp.751-769, 1998.

[29] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn and R. T. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*. Vol.22, No.3, pp.401-410, 2003.

[30] M. Muller, C. Wojtan and T. Brochu. Liquid simulation with mesh-based surface tracking. *SIGGRAPH 2011 Course Notes*, 2011.

[31] A. Shakibaeinia and Y. C. Jin. A weakly compressible MPS method for modeling of open-boundary free surface flow. *Int. J. Numer. Meth. Fluids*, Vol.63, pp.1208-1232, 2010.

[32] B.B. Mandelbrot. The fractal geometry of nature. *W. H. Freeman and Company*, New York, 1981.

[33] H. O. Peitgen and D. Saupe. The Science of fractal image. *Springer Verlag*, 1988.

Gantulga Tsedendorj is a PhD candidate at the Department of Computer Science, Iwate University. His research interests include Mathematical Modeling and Physically-based Computer Simulation and Animation. He studied at the Faculty of Computational Mathematics and Cybernetics, Moscow State University, 1990-1994. He received his BS and MS degrees in Applied Mathematics from the National University of Mongolia and University of Colorado at Denver in 1997 and 2002, respectively. He has worked previously as an instructor at the Ulaanbaatar University, 1997-2003 and as a researcher at the Institute of Mathematics, National University of Mongolia, 2003-2004. He currently is a lecturer of Computational Mathematics at the School of Mathematics and Computer Science, National University of Mongolia.

Norishige Chiba is currently a Professor in the Department of Computer Science at Iwate University. His research interests include Computer Graphics, Laser Graphics and Interactive Graphics. He received a BE in electrical engineering from Iwate University and an ME and DE in information engineering from Tohoku university in 1975, 1981 and 1984, respectively. He worked at Nippon Business Consultant Co., Ltd from 1975 to 1978. He was a research associate in the Department of Communication Engineering at Tohoku University from 1984 to 1986, an associate professor of the Department of Computer Science at Sendai National College of Technology from 1986 to 1987 and an associate professor of the Department of Computer Science at Iwate University from 1987 to 1991. He is a member of The Society for Art and Science, IPS, IEEE and ACM.

Screenshots of the animation experiments

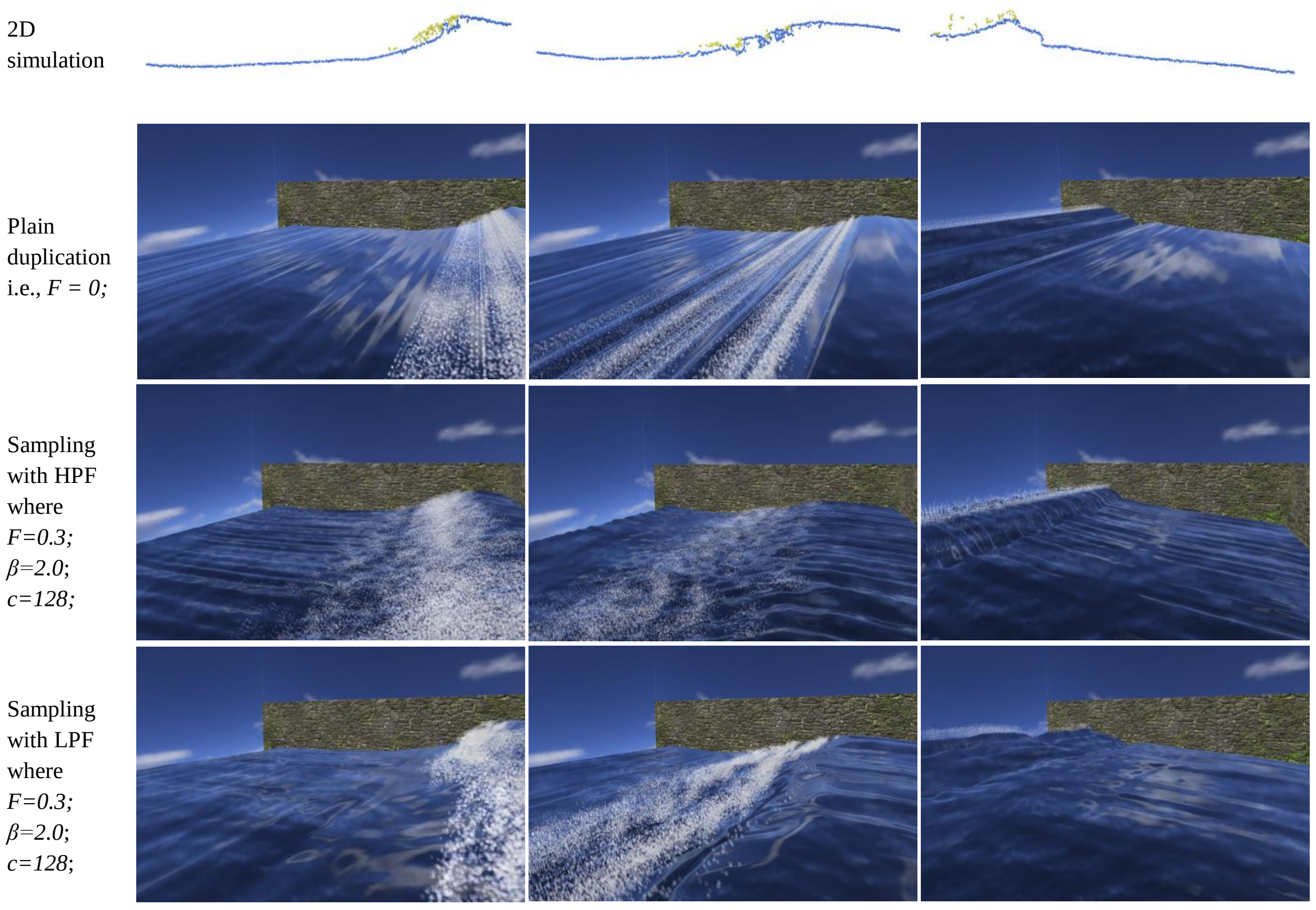
2D simulation
Plain duplication i.e., *F = 0;*
Sampling with HPF where *F=0.3; β=2.0; c=128;*
Sampling with LPF where *F=0.3; β=2.0; c=128;*

Figure 13. *Animation of a surging wave with splashes.*

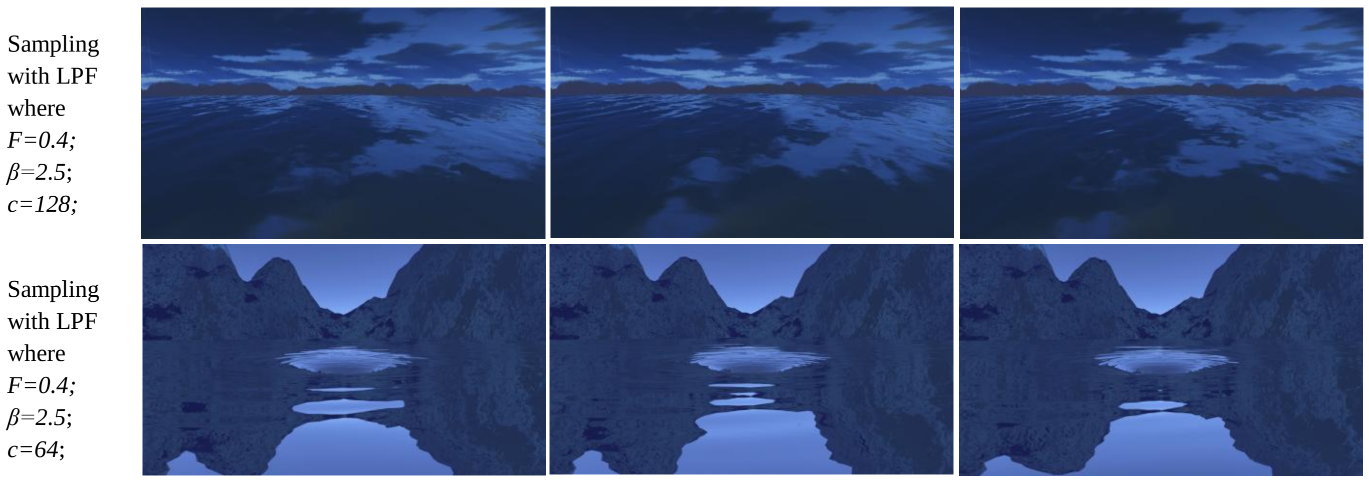
Sampling with LPF where *F=0.4; β=2.5; c=128;*
Sampling with LPF where *F=0.4; β=2.5; c=64;*

Figure 14. *Animations of ripple and whirlpool.*