2005－CG－118　(12)
2005／2／7

Rendering Breaking Waves Efficiently

Wang Qiang[1,2], Toru Suzuki[3], Tadahiro Fujimoto[3] and Norishige Chiba[3]

[1]Invited Research Fellow of National Institute of Information and Communication Technology
[2]Invited Research Fellow of Iwate University
[3]Department of Computer Science, Faculty of Engineering, Iwate University
(wangqiang@cg.cis.iwate-u.ac.jp, fujimoto@cis.iwate-u.ac.jp, nchiba@cis.iwate-u.ac.jp)

In this paper we propose a particle-based scheme for modeling and rendering the phenomenon of breaking waves. Firstly, MPS in 2D is used, and then its result is expanded to three-dimension representation by giving motion variation using fBm. Secondly, the surface of water body is reconstructed from the outlines of 2D slices. Finally, the splashing effect is computed according to the properties of the particles. The result of our experiments show that the proposed method combining the MPS and fBm describes the behavior of breaking waves successfully, and our method produces realistic effects of breaking waves effectively and efficiently.

砕け波の効率的なレンダリング法

王 強 [1,2]　　鈴木 亨 [3]　　藤本 忠博 [3]　　千葉 則茂 [3]

[1]情報通信研究機構招聘研究員　[2]岩手大学客員研究員　[3]岩手大学工学部情報システム工学科

粒子ベースによる砕け波のモデリングならびにレンダリング法を提案する．まず，ＭＰＳ法により求めた２次元の砕け波の粒子運動を fBm を用いて３次元に拡張する．そして，２次元の粒子運動の外形に基づき波の表面を構築する．また，粒子の属性に応じて飛沫を表現する．実験の結果，リアルな砕け波の振る舞いが効率的に生成されることが確認できた．

1. Introduction

Breaking wave is an interesting natural phenomenon. Many literary work including fictions, films and poems describe its beauty and pageantry. Hokusai, Japan's best known plastic artist, had a famous painting "In the Hollow of a Wave off the Coast at Kanagawa" (Figure 1) [Breen04]. It gives us a wonderful observation of the breaking wave.

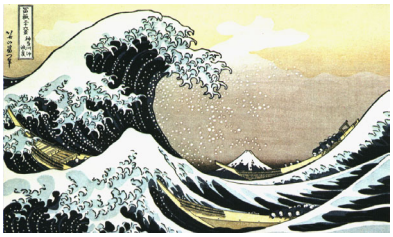

Figure 1. "The Wave" by Hokusai

Breaking wave is also one of the most complex natural phenomena. It is still an opening challenging problem to produce realistic image of breaking wave in computer graphics. The solution usually contains two components. One is to simulate the behavior of the wave, and the second is to render the scene. Several methods for representing breaking waves have been proposed: methods based on sinusoidal and trochoidal functions [Fournier86, Jeschke03, Peachey86]; a method using water current simulation [Enright02].

Fractional Brownian motion (fBm) is also known as $1/f^{\beta}$ noise, which is noise with a power spectrum inversely proportional to frequency f to the power of β. The value of β determines a noise correlation. When the value of β is 0, it corresponds to white noise; when β is 2, it represents normal Brownian motion. It is well known that this fBm is observed in many natural phenomena, such as terrain shapes, fluid phenomena, and organic phenomena, what is called "waving phenomena" [Mandelbrot77].

In this paper, we proposed a method to produce realistic breaking wave images. The MPS (Moving Particle Semi-implicit) in 2D [Koshizuka95] and fBm is combined for the simulation of breaking wave. MPS method is used to produce a 2D slice of the wave, and then its result is expanded to 3D by giving motion variation using fBm.

We can then reconstruct the surface of the water body from the outlines of the 2D slices. Surface reconstruction from contours of parallel slices has a long history. Many researchers have good result [Christiansen78, Ekoule91, Meyers92], and can treat complicated cases such as concave contours.

Splashes and foam is also an important feature of the breaking wave. Up to now, few research work of ocean waves concerns with this problem. Takahashi et al. have recently proposed a splash and form representation method that is visually extremely effective [Takahashi03]. Mihalef et al. proposed a scheme of animation and control of breaking wave [Mihalef04]. In this paper we generate the splashes accompanying with the wave motion according to the properties of the particles.

To render realistic images, the environment mapping to the surface is also considered. We generated the environment mapping and blending effects on the water surface using Cg shader.

2. Particle-based Animation of Breaking Waves

Physical model is an important component of fluid simulations. Most of the previous works are based on the Navier-Stokes equation. There are different solvers for simulation in Computer Graphics. J. Stam proposed a stable but not accurate method solver. Takahashi et al. use CIP method to solve the equations [Takahashi03].

We employ the MPS method [Koshizuka95] to solve the physical model of the simulation phase.

2.1 Moving Particle Semi-implicit Method (MPS)

The MPS method used in the proposed method is a Lagrange type simulation method using particles, and handles incompressible flow. The right side of the Navier-Stokes equation in equation (1), which is a governing equation of fluid, is made up of a pressure term, a viscosity term and an external force term.

$$\frac{du}{dt} = -\frac{1}{\rho}\nabla P + \nu\nabla(\nabla u) + f \qquad (1)$$

Here, t is time, u is velocity, ρ is density, P is pressure, ν is dynamic coefficient of viscosity, and f is external force. The MPS method discretizes this equation by using interaction between particles. The interaction between particles is modeled based on the weight function $w(r)$ below.

$$w(r) = \begin{cases} \frac{r_e}{r} - 1 & 0 \le r < r_e \\ 0 & r_e < r \end{cases} \qquad (2)$$

Here, r is a distance between two particles, and r_e is a range in which the interaction between particles extends. The gradient model at the position of a particle i is as follows.

$$\langle \nabla\phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\phi_j - \phi_i}{\left|\vec{r_j} - \vec{r_i}\right|^2} \left(\vec{r_j} - \vec{r_i}\right) w\left(\left|\vec{r_j} - \vec{r_i}\right|\right) \qquad (3)$$

For a physical quantity φ, this equation means averaging gradient vectors $\left(\phi_j - \phi_i\right)\left(\vec{r_j} - \vec{r_i}\right)\Big/\left|\vec{r_j} - \vec{r_i}\right|^2$ between a particle i and particles j around the particle i using the weight function w. Here, d is space dimension, and n0 is a fixed value for particle density. The Laplacian model at the position of a particle i is as follows.

$$\left\langle \nabla^2\phi \right\rangle_i = \frac{2d}{n^0\lambda} \sum_{j \neq i} \left(\phi_j - \phi_i\right) w\left(\left|\vec{r_j} - \vec{r_i}\right|\right) \qquad (4)$$

This equation means providing physical quantity of particles j around particle i to the particle i using the weight function w. Here, λ is a coefficient for adjusting the variance of the distribution to analytical solution.

2.2 Two-Dimensional Simulation

In the proposed method, first of all, two-dimensional simulation of waves washing against a beach or a wharf is carried out using the MPS method. The waves are generated using a wave making method called "piston type" model with a wave making plate, as shown in Figure 2. In order to simplify the implementation of the simulation, the beach, wharf, and wave making plate are also represented using particles that are solidified. Figure 3 shows simulation results of waves washing up on a beach, and waves washing against a wharf. Table 1 shows the numbers of particles used and computation time. The number of simulation steps was 20,000, and the computing environment used was a 2.6 GHz Pentium 4 CPU and 512 M Byte of memory.

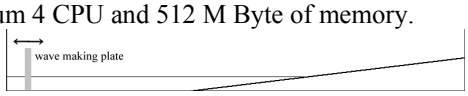

Figure 2. Piston type wave making method.

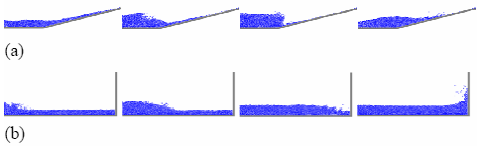

Figure 3. Frame sequences of animations of breaking waves generated by two-dimensional simulation. (a) Waves washing up on a beach. (b) Waves washing against a wharf.

Table 1. The numbers of particles used and computation time for two-dimensional simulation of breaking waves.

	total number of particles	number of fluid particles	Number of wall particles	computation time
beach	2903	1625	1278	4132 sec
wharf	3993	2651	1342	6076 sec.

2.3 Expansion to Three-Dimensional Animation

In the proposed method, basically, a three-dimensional animation is obtained by arranging n two-dimensional simulation results in the direction orthogonal to wave advancing direction, as shown in Figure 4. However, if the arranged two-dimensional results are completely the same, the resulting three-dimensional motion has an unnatural impression. Therefore, in the proposed method, we utilize two-

dimensional fBm to remove the uniformity of the three-dimensional wave motion as described below.

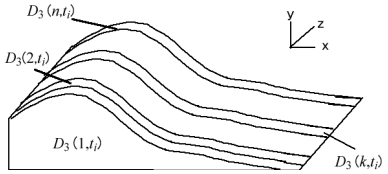

Figure 4. Expansion of two-dimensional simulation results to three dimensions.

First, $D_2(s)$ denotes a set of position data of particles obtained at each computation step s = 0, 1, 2, ... of the two-dimensional simulation. A two-dimensional animation is usually generated by displaying $D_2(s_i)$ obtained at sampling time s_i for each fixed increment Δs at frame time t_i. Here, if a three-dimensional animation is obtained by arranging n sets of the same $D_2(s_i)$, as described above, a set of position data $D_3(k, t_i)$ of particles in line k = 1, 2, 3, …, n on the three-dimensional space at frame time t_i for each fixed increment Δt becomes as follows (Figure 4).

$$D_3(k, t_i) = D_2(s_i),\ k = 1,2,\ldots,n \quad (5)$$

$$t_0 = 0,\ t_{i+1} = t_i+\Delta t,\ i = 0,1,2\ldots \quad (6)$$

$$s_0 = 0,\ s_{i+1} = s_i+\Delta s,\ i = 0,1,2\ldots \quad (7)$$

In order to produce three-dimensional natural motion in $D_3(k, t_i)$, the proposed method samples D_2 at different sampling time $s_{k,i}$ for each line k, not at the same sampling time s_i for all lines k as described above. Specifically, $D_3(k, t_i)$ for line k at frame time t_i is obtained using the following equations.

$$D_3(k, t_i) = D_2(s_{k,i}),\ k = 1,2,\ldots,n \quad (8)$$

$$t_0 = 0,\ t_{i+1} = t_i+\Delta t,\ i = 0,1,2\ldots \quad (9)$$

$$s_{k,i} = 0,\ s_{k,i+1} = s_{k,i}+\Delta s_{k,i},\ i = 0,1,2\ldots \quad (10)$$

$$\Delta s_{k,i}=f_d(\Delta s, \mathrm{Noise}(k, t_i)),\ i = 0,1,2\ldots \quad (11)$$

Here, Noise(k, t) is a two-dimensional fBm noise function. The section of this function for each k is one-dimensional fBm noise function, which has correlation to one another. The function f_d rounds up or down the value of Δs Noise(k, t_i) to a discrete value $\Delta s_{k,i}$ so as to make $s_{k,i}$, i = 0, 1, 2, …, sampling time at which a data exists in D_2. This mechanism appropriately changes the sampling interval $\Delta s_{k,i}$ for each line k and each frame time ti, and results in realizing natural three-dimensional motion. In implementation, the function Noise(k, t) only has a finite length with respect to t. Therefore, in order to perform sampling over a long time period t, the function is used cyclically. Besides, in order to acquire an appropriate sampling interval in equation (11), the function is given the condition that $0.75 \le Noise(k,t) \le 1.25$.

3. Surface Reconstruction and Rendering of the Water Body

The geometric representation of the fluid body is another component of fluid simulation. Triangular mesh is usually used. Many approaches, such as level set based methods [Enright02, Foster01, Takahashi03], use modified marching cubes to extract the surface of the triangular surface of the water body.

We get the triangular mesh of the breaking wave by tiling triangle strips from the outlines of 2D slices. This process is divided into two steps: extracting the outlines of the 2D simulation; constructing the smooth surface from the 2D outlines.

3.1 Outline Extraction of the 2D Simulation

To extract the outlines of the 2D MPS simulation result, firstly we have to project the simulation to a grid. A slice of 2D MPS of particles is demonstrated below.

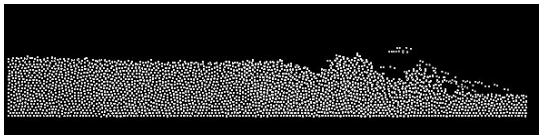

Figure 5. A slice of 2D simulation.

In the initial state of the MPS computation, the particles are put into a grid of “Piston Type” Model. We construct a blank image with the same resolution as this grid’s. The image is filled initially with background color.

Each particle is projected onto this image. If a particle is fallen among the four neighboring grid points, then the four pixels is marked to be foreground pixel. Figure 6 shows the projected resulting image of the slice of Figure 5.

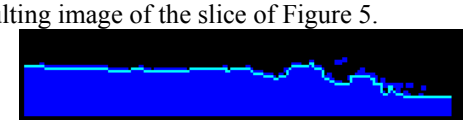

Figure 6. Projected Image of 2D Simulation.

Then we can extract the outline using some technique of image processing. In Figure 6, the outline is drawn in a sky blue color. The outline is compound of successive pixels (grid points) in a fixed direction, say anti-clockwise order. Moreover, the extracted outline is not a contour which is closed. We just take the part of the contour on the top face of the water surface, neglecting the parts of the side faces and the bottom face.

3.2 Surface Reconstruction from the 2D Outlines

The slices of 2D MPS simulation are parallel, so they are arranged along the z-axis, and each slice is given a fixed z-value. The tiling process is performed on the outlines pair by pair. For each pair of outline, it is partitioned into segments, and then we tile triangle strips segment by segment.

3.2.1 Partition of Outline Pair

Given an outline $O_j=\{P_i, i=j_1,\ldots,j_M\}$, it is defined on a x-y plane, and all the vertices have a fixed z-coordinate, say z_j.

As shown in the following figure, the outline may have more than one intersection points with the vertical straight line $x=x_i$. We use $F(x_i, z_j)$ to denote this number.

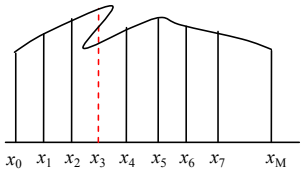

Figure 7. An outline on x-y plane.

For two outlines $O_j=\{P_i, i=j_1,\ldots, j_M\}$ and $O_j=\{P_i, i=j'_1,\ldots, j'_N\}$, we can divide them into $l(l>0)$ successive segment pairs along the x-axis direction:

$$\begin{aligned}
&\{P_{j_1},\cdots,P_{k_1}\}\{P_{j'_1},\cdots,P_{k'_1}\}\\
&\{P_{k_1},\cdots,P_{k_2}\}\{P_{k'_1},\cdots,P_{k'_2}\}\\
&\cdots\\
&\{P_{k_{l-1}},\cdots,P_{j_M}\}\{P_{k'_{l-1}},\cdots,P_{j'_N}\}\\
&x_{k_i}=x'_{k_i},\quad i=1,\cdots,l-1
\end{aligned}$$

where $F(x_{k_i},z_j)=1$ and $F(x_{k'_i},z_{j'})=1$, and (x_i, y_i, z_i) is the coordinate of the vertex P_i.

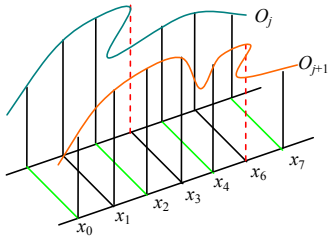

Figure 8. Partition of a pair of outlines.

As shown in the above figure, the outline can be partitioned into segments along x-axis direction: x_0-x_2, x_2-x_4, x_4-x_7…

3.2.2 Tiling Triangular Strip

After the outlines are partitioned into segments along x-axis, we then construct the triangular surface segment by segment.

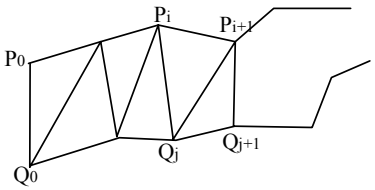

Figure 9. Tiling of triangles between two outlines.

Given two outline segments $O_1=\{P_i, i=1,\ldots, M\}$ and $O_2=\{Q_i, i=1,\ldots, N\}$ and a weight function $w(V_1, V_2)$, where V_1 and V_2 are two successive points on an outline, we tile a triangle strip between O_1 and O_2. We define $W(P_i,P_{i+k})=\sum_{j=i}^{i+k-1} w(P_j,P_{j+1})$.

In the initial step, we set $i=j=0$.

There are many tiling criteria [Christiansen78, Ekoule91, Meyers92]. Here we choose the following.

If the inequality (12) is satisfied, then the triangle $P_iP_{i+1}Q_j$ is added, otherwise the triangle $P_iQ_jQ_{j+1}$ is added.

$$\begin{aligned}&|W(P_0, P_i) + w(P_i, P_{i+1}) - W(Q_0, Q_j)| <\\ &|W(Q_0, Q_j) + w(Q_j, Q_{j+1}) - W(P_0, P_i)|\end{aligned} \qquad \textbf{(12)}$$

The weight function plays a key role in the tiling process. A common weight function is the normalized distance between two vertices:

$$w(V_1, V_2) = |V_1V_2| / L.$$

where L is the total length of the outline segment, and $|V_1V_2|$ means the length of line segment V_1V_2.

But this function can not deal with concave outlines. It may produce distorted surface in concave cases. In the following section we describe the algorithm for the calculation of the weight function.

3.2.3 Calculation of the Weight Function

For the non-convex case, we have to deal with the concave section on the outline. We use the method in [Ekoule91] to calculate the weight of the edges.

Given a contour $C=C^0=\{P_i, i=1,\ldots, M\}$, H^0 is the convex hall of C^0. P_{i1} and P_{j1} is two successive vertices in H^0 but not successive in C^0. $C^1_{(i_1,j_1)}$ denotes the set of points on the contour between P_{i1} and P_{j1}, and this set is defined to be 1-order concave section. In Figure 10, we have $C^1_{(3,10)}=\{P_3,P_4,\ldots,P_{10}\}$.

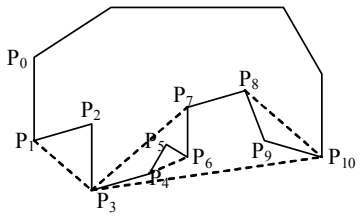

Figure 10. The decomposition of a non-convex contour.

For a *n*-order concave section $C^{n}_{(i_1,j_1)(i_2,j_2)\ldots(i_n,j_n)}$, and its convex-hull H^n. If they are the same, then the decomposition terminates. If not, then the decomposition continues and we get the (*n*+1)-order concave section.

After the decomposition of the contour, the next step is to project each vertex onto its convex hull.

For a n-order concave section $C^{n}_{(i_1,j_1)(i_2,j_2)\ldots(i_n,j_n)}$, we first project to the vertices in this section to $P_{i_n}P_{j_n}$.

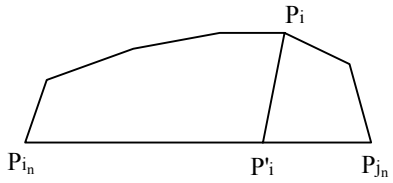

Figure 11. The projection for concave section.

For a point $P_i, i \in (i_n, j_n)$, we project it to a point P'_i, which is on the line segment $P_{i_n}P_{j_n}$. Its coordinate (x'_i, y'_i) is calculated as follows:

$$\begin{cases} x'_i = x_i + R_i(x_{j_n} - x_{i_n}) \\ y'_i = y_i + R_i(y_{j_n} - y_{i_n}) \end{cases} \quad (13)$$

Where R_i is given as:

$$R_i = \frac{\sum_{k=i_n}^{i-1} d(P_k, P_{k+1})}{\sum_{k=i_n}^{j_n-1} d(P_k, P_{k+1})} \quad (14)$$

The above process can be continued recursively, until the vertices are projected onto the initial convex-hull H^0.

Assuming that P_i is finally projected to $\overline{P}_i$ on H^0, we calculate the weight as follows:

$$w(P_i, P_{i+1}) = \left|\overline{P}_i\overline{P}_{i+1}\right|$$

3.3 Smoothing the Reconstructed Surface

To delimitate the noise of particle simulation, we perform some denoising operation.

The denoising operation performs in two directions. One is to smooth outlines, and the other is to reduce the bumping between the adjacent outlines.

The first pass of smoothing is done after the outline is computed. We set the coordinate of the vertex as the weighted sum of the coordinated of itself and the proceeded and succeeded vertices.

The second pass of denoising operation is performed between adjacent outlines. For each vertex on the outline, we find its neighboring vertices on adjacent outlines, and then we set the new coordinate of this vertex as weighted sum of the coordinates of the vertices in the neighborhood.

4. Splash Rendering

Splashing effect is an important feature of a scene of breaking wave. Currently, few research results are known about the rendering splashing effect. Takahashi et al. have recently proposed a splash and foam representation [Takahashi03]. In this paper, we render the splash as follows.

A particle around which the density of particles is smaller than a predefined threshold value in two-dimensional simulation is determined to be an original splash particle. Then, so as to effectively visualize splashes, this particle is replaced with several new splash particles, which are rendered in white.

The new splash particles have a lifespan. After displaying them during the lifespan, they return to the original particle, which is rendered in original water color.

If the threshold value, the number of new splash particles, and the lifespan above are fixed, some noticeable defects occur due to the correlation of fBm; for example, white splash particles stretch out in a belt-shape. In order to avoid such visual artifacts, techniques below are used.

- The threshold value is changed randomly for each line.
- The number of new splash particles is changed according to the speed of the original particle.
- The lifespan is selected randomly.

In order to efficiently generate a long period of animation using fewer simulation steps, a proper data segment, which contains a few typical wave pushing and washing up motions, of a short interval $<s_a, s_b>$ in a two-dimensional simulation result is used cyclically. In this case, the segment is selected such that the particle configurations of $D_2(s_a)$ and $D_2(s_b)$ look similar. Also, if the fBm noise function Noise(k, t) is simply used cyclically with respect to t, the sway of the wave crest gradually becomes too large due to the difference in the sampling time $s_{k,i}$ of equation (25) given by accumulating the sampling intervals $\Delta s_{k,i}$ of equation (26) between lines k. In order to avoid this, when t is used cyclically, k is also used cyclically such that $k = k + \Delta k$. In generating the animation examples of Figure 11, Δk was set to 20.

5. Shading of the Scene

To render the realistic effect of the breaking wave efficiently, we use Cg shader to compute the environment mapping, reflective and refractive effect of the water body.

6. Conclusions and Future Work

We have proposed a scheme for generation and rendering of the breaking wave. The fBm together with MPS system are used to simulate the motion behavior of wave. In the rendering phase, we reconstruct the surface of the wave from the outlines of 2D slices of MPS simulation. The splash is generated according to properties of the particles.

Figure 12 shows several frames of the animation of breaking wave. The images show the water surface of the water body, splash effects and environment mapping.

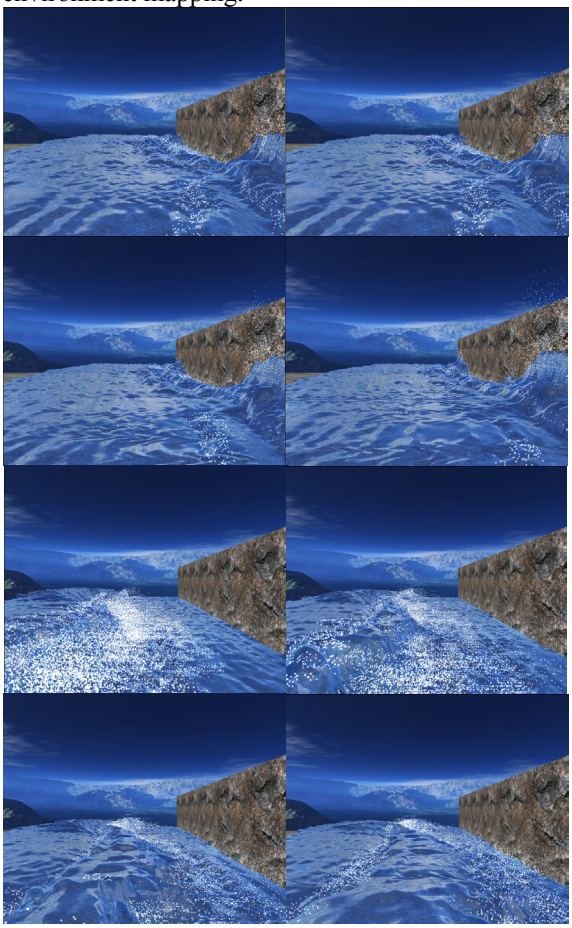

Figure 12. Frames of the animation of breaking wave.

The average time of reconstructing and rendering a frame is about 0.45s. The grid size of the simulation is 256×128, and it is expand to 256 slices. This scale is much larger than those using 3D simulations [Enright02, Takahashi03]. The computing environment used was a 2.0 GHz Pentium 4 CPU and 512 M Byte of memory.

In the experiments, we use fixed number of particles. In the future, we may explore the method to generate break waves using “infinite” particles.

Acknowledgements

The authors would like to express thanks to Prof. Koshiduka, The University of Tokyo, for his source code of 2D MPS. This work is supported partly by National Institute of Information and Communication Technology.

References:

[Breen04] Jim Breen's Ukiyo-E Gallery – Hokusai, http://www.csse.monash.edu.au/~jwb/ukiyoe/hokusai.html.

[Christiansen78] Christiansen, H. N., Sederberg, T. W., "Conversion of Complex Contours Line Definition into Polygonal Element Mosaics", Computer Graphics, 12(2), 1978, pp. 187-192.

[Ekoule91] Ekoule A. B., F. C. Peyrin, C. L. Odet, "A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours", ACM Transactions on Graphics, Vol. 10, No. 2, April 1991, pp. 182-199.

[Enright02] Enright, S. Marschner, and R. Fedkiw, Animation and Rendering of Complex Water Surfaces, Proceedings of SIGGRAPH 2002, pp.736-744, 2002.

[Foster01] N. Foster and R. Fedkiw. Practical Animation of Liquids, pages 23-30. ACM SIGGRAPH 2001, 2001.

[Fournier86] A. Fournier and T. Reeves, A simple model of ocean waves, Proceedings of SIGGRAPH 1986, vol.20, pp.75-84, 1986.

[Jeschke03] S. Jeschke, H. Birkholz, and H. Schmann, A Procedural Model for Interactive Animation of Breaking Ocean Waves, WSCG'2003 POSTERS Proceedings, 2003.

[Koshizuka95] S. Koshizuka, H. Tamako, and Y. Oka, A Particle Method for Incompressible Viscous Flow with Fluid Fragmentation, Comput. Fluid Dynamics, J.4, pp.29-46, 1995.

[Mandelbrot77] B. B. Mandelbrot, The fractal geometry of nature, W. H. Freeman and Company, New York, 1977.

[Mihalef04] Viorel Mihalef, Dimitris Metaxas, Mark Sussman, Animation and control of breaking waves, Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, August 27-29, 2004, Grenoble, France.

[Meyers92] Meyers, D., Skinner, S., "Surfaces from Contours", ACM Transactions on Graphics, Vol. 11, No. 3, 1992, pp. 228-258.

[Peachey86] D. R. Peachey, Modeling waves and surf, Proceedings of SIGGRAPH 1986, pp.65-74, 1986.

[Stam99] J. Stam. Stable Fluids, pages 121-128. ACMSIGGRAPH 99, 1999.

[Takahashi03] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki, Realistic Animation of Fluid with Splash and Foam, EUROGRAPHICS 2003, pp.391-400, 2003.