Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# Efficient rendering of breaking waves using MPS method*

WANG Qiang†[1], ZHENG Yao[1], CHEN Chun[1], FUJIMOTO Tadahiro[2], CHIBA Norishige[2]

([1]*School of Computer Science, Zhejiang University, Hangzhou 310027, China*)

([2]*Department of Computer Science, Faculty of Engineering, Iwate University, Morioka 020-08550, Japan*)

†E-mail: wangqiang@cad.zju.edu.cn

Received Aug. 31, 2005; revision accepted Nov. 28, 2005

**Abstract:** This paper proposes an approach for rendering breaking waves out of large-scale of particle-based simulation. Moving particle semi-implicit (MPS) is used to solve the governing equation, and 2D simulation is expanded to 3D representation by giving motion variation using fractional Brownian motion (fBm). The waterbody surface is reconstructed from the outlines of 2D simulation. The splashing effect is computed according to the properties of the particles. Realistic features of the wave are rendered on GPU, including the reflective and refractive effect and the effect of splash. Experiments showed that the proposed method can simulate large scale breaking waves efficiently.

**Key words:** Moving particle semi-implicit (MPS), Particle-system, Surface reconstruction
**doi:**10.1631/jzus.2006.A1018 **Document code:** A **CLC number:** TP391

## INTRODUCTION

Breaking wave is an interesting natural phenomenon. Many literary work including fiction, films and poems describe its beauty and pageantry. Fig.1 shows a wonderful scene of breaking wave.

**Fig.1 A portrait of breaking wave**

Breaking wave is also one of the most complex natural phenomena. It is still an open challenging problem to produce realistic image of breaking wave in computer graphics. Breaking waves are traditionally modelled using sinusoidal and trochoidal functions (Fournier and Reeves, 1986; Jeschke *et al*., 2003; Peachey, 1986). For more visually realistic effect sea waves, physics-based simulation is developed (Takahashi *et al*., 2003; Mihalef *et al*., 2004; Fujimoto *et al*., 2005).

Physics-based simulation of breaking waves is based on governing equation of fluid, Navier-Stokes equation. There are several kinds of solvers of this equation for simulating purpose in computer graphics. Stam (1999) developed a stable but inaccurate method to solve the equation. His method is suitable for simulation of gaseous phenomenon. Physically based simulation of water movement is also very successful and has achieved impressive results (Foster and Fedkiw, 2001; Enright *et al*., 2002). Takahashi *et al*.(2003) used CIP method to solve the equations. Song *et al*.(2005) proposed a method to improve the accuracy of the method proposed by Stam (1999).

There also exist particle-based methods for simulation of fluids. Stam and Fiume (1995) described fire and gaseous phenomena using "smoothed

* Project partly supported by the National Institute of Information and Communication Technology (NICT), Japan

particle hydrodynamics" (SPH). In SPH, the fluid is modelled as a collection of particles with a smoothed potential field. Premoze *et al*.(2003) used moving particle semi-implicit (MPS) method to simulate incompressible multiphase fluids.

Unfortunately, existing methods are computationally expensive and slow, so large-scale problems requiring large grids are currently impractical to simulate.

In this paper, we propose a method to efficiently produce realistic images of breaking wave. Our goal is to render several frames in 1 s. Currently MPS in 3D is still computationally expensive to use. We use MPS in 2D (Koshizuka *et al*., 1995) combined with fractional Brownian motion (fBm) (Mandelbrot, 1977) to simulate the breaking waves. 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, which has been successfully used in modelling natural phenomena.

The waterbody surface mesh is then reconstructed from the outlines of the 2D slices. It has been studied since the 1970's for the problem of surface reconstruction from contours of parallel slices (Christiansen and Sederberg, 1978; Ekoule *et al*., 1991; Meyers and Skinner, 1992). The polygonal surface is tiled from the contours. In the case of multiple-contour in one slice, the problems of correspondence and branching have to be solved. Ekoule *et al*.(1991) introduced the method of decomposition and projection to deal with concave contours during the tiling process.

Splashes and foam are also important breaking wave features. Up to now, few research work on ocean waves concerning this problem exist. Takahashi *et al*.(2003) proposed a splash and foam representation method that is visually extremely effective. In this paper we generate the splashes accompanying wave motion according to the properties of the particles.

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

The rest of this paper is organized as follows: in Section 2 we discuss the physical model of fluid simulation and some details of the solver; surface reconstruction from 2D outlines are described in Section 3; in Section 4, we treat the problem of splash rendering; experiments and comments are given in Section 5, and finally we summarize the conclusions and future work in Section 6.

## PARTICLE-BASED ANIMATION OF BREAKING WAVES

In previous work in simulation of waves, Navier-Stokes equation was solved using grid-based methods. There are different types of solvers. In this paper, we employ a particle-based method, the MPS method (Koshizuka *et al*., 1995), to solve the physical model.

To render the wave efficiently, we use 2D MPS to simulate the wave motion, and add fBm noise on 2D MPS to make the wave motion natural (Fujimoto *et al*., 2005).

### Governing equations of fluid

The motion of incompressible fluid can be described by the following equations:

$$\frac{\mathrm{d}\rho}{\mathrm{d}t}=0, \tag{1}$$

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}=-\frac{1}{\rho}\nabla P+\nu\nabla(\nabla\boldsymbol{u})+\boldsymbol{f}. \tag{2}$$

Eq.(2) is the Navier-Stokes equation describing the behavior of fluid. Eq.(1) states that the density $\rho$ is constant. The right side of the Navier-Stokes equation in Eq.(2), which is a governing equation of fluid, is made up of a pressure term, a viscosity term and an external force term. Here, $t$ is time, $\boldsymbol{u}$ is velocity, $\rho$ is density, $P$ is pressure, $\nu$ is dynamic coefficient of viscosity, and $\boldsymbol{f}$ is external force.

### MPS method

The MPS method used in the proposed method is a Lagrange type simulation method using particles. It discretizes Eq.(2) by using the interaction between particles. The interaction between particles is modelled based on the weight function $w(r)$ below:

$$w(r)=\begin{cases}\dfrac{r_\mathrm{e}}{r}-1, & 0\le r<r_\mathrm{e},\\ 0, & r\ge r_\mathrm{e}.\end{cases} \tag{3}$$

Here, $r$ is distance between two particles, and $r_e$ is the range over which the interaction between particles extends.

The gradient model at the position of a particle $i$ is as follows:

$$\langle \nabla \phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \frac{\phi_j - \phi_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2} (\boldsymbol{r}_j - \boldsymbol{r}_i) w(|\boldsymbol{r}_j - \boldsymbol{r}_i|). \quad (4)$$

For a physical quantity $\phi$, this equation means averaging gradient vectors $(\phi_j - \phi_i)(\boldsymbol{r}_j - \boldsymbol{r}_i)/|\boldsymbol{r}_j - \boldsymbol{r}_i|^2$ between a particle $i$ and particles $j$ around the particle $i$ using the weight function $w$. Here, $d$ is space dimension, and $n^0$ is a fixed value for particle density. The Laplacian model at the position of a particle $i$ is as follows:

$$\langle \nabla^2 \phi \rangle_i = \frac{2d}{n^0 \lambda} \sum_{j \neq i} (\phi_j - \phi_i) w(|\boldsymbol{r}_j - \boldsymbol{r}_i|). \quad (5)$$

This equation means providing physical quantification of particles $j$ around particle $i$ using the weight function $w$. Here, $\lambda$ is a coefficient for adjusting the variance of the distribution in the analytical solution.

The particle number density can be computed as follows:

$$\langle n \rangle_i = \sum_{j \neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|). \quad (6)$$

The MPS algorithm can be outlined as follows:

**Algorithm 1** The MPS algorithm

Set the initial value of velocity $u^0$ and position $r^0$.
for $n$=1 to maximal step $N$
  Compute the temporary particle velocities $u^*$ and positions $r^*$;
  Compute particle number density $n^*$ using temporary particle locations $r^*$;
  Set up and solve Poisson pressure equation;
  Compute velocity correction $u'$ from the pressure equation;
  Compute new particle positions and velocities:
    $u^{n+1}=u^*+u'$
    $r^{n+1}=r^*+u'\mathrm{d}t$
end for

The Poisson equation for pressure is as follows:

$$\langle \nabla p^{n+1} \rangle_i = -\frac{\rho}{\mathrm{d}t} \frac{\langle n^* \rangle_i - n^0}{n^0}. \quad (7)$$

After the Poisson equation is solved, the correction of velocity $u'$ is computed:

$$\boldsymbol{u}' = -\frac{\mathrm{d}t}{\rho} \langle \nabla p^{n+1} \rangle. \quad (8)$$

We can then get the velocity and position of the next step.

## 2D simulation

2D 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 Fig.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. Fig.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 20000, and the computing environment used was a 2.6 GHz Pentium IV CPU and memory of 512 MB.

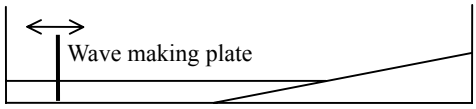
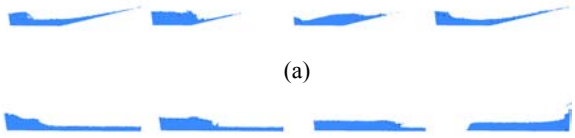



**Fig.2 Piston type wave making method**



(a)

(b)

**Fig.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 Numbers of particles used and computation time for 2D simulation of breaking waves**

| Parameter | Beach | Wharf |
|---|---|---|
| Total number of particles | 2903 | 3993 |
| Number of fluid particles | 1625 | 2651 |
| Number of wall particles | 1278 | 1342 |
| Computation time (s) | 4132 | 6076 |

**Expansion to 3D animation**

In the proposed method, basically, a 3D animation is obtained by arranging $n$ 2D simulation results in the direction orthogonal to wave advancing direction, as shown in Fig.4. However, if the arranged 2D results are completely the same, the resulting 3D motion gives an unnatural impression. Therefore, in the proposed method, we utilize 2D fBm to remove the uniformity of the 3D wave motion as described below.

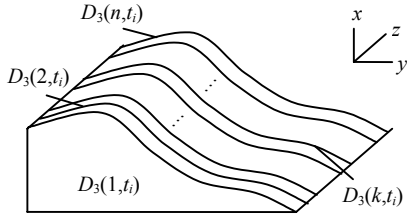



**Fig.4 Expansion of 2D simulation results to 3D**

First, let $D_2(s)$ denote a set of particles position data obtained at each computation step $s$=0, 1, 2, ... of the 2D simulation. A 2D animation is usually generated by displaying $D_2(s_i)$ obtained at sampling time $s_i$ for each fixed increment $\Delta s$ at frame time $t_i$. Here, if a 3D animation is obtained by arranging $n$ sets of the same $D_2(s_i)$, as described above, a set of particles position data $D_3(k,t_i)$ in line $k$=1, 2, 3, …, $n$ on the 3D space at frame time $t_i$ for each fixed increment $\Delta t$ becomes as follows (Fig.4):

$$D_3(k,t_i)=D_2(s_i),\ k=1,\ 2,\ \ldots,\ n \tag{9}$$

$$t_0=0,\ t_{i+1}=t_i+\Delta t,\ i=0,\ 1,\ 2,\ \ldots \tag{10}$$

$$s_0=0,\ s_{i+1}=s_i+\Delta s,\ i=0,\ 1,\ 2,\ \ldots \tag{11}$$

In order to produce 3D natural motion in $D_3(k,t_i)$, the proposed method samples $D_2$ are obtained 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:

$$D_3(k,t_i)=D_2(s_{k,i}),\ k=1,\ 2,\ \ldots,\ n \tag{12}$$

$$t_0=0,\ t_{i+1}=t_i+\Delta t,\ i=0,\ 1,\ 2,\ \ldots \tag{13}$$

$$s_{k,0}=0,\ s_{k,i+1}=s_{k,i}+\Delta s_{k,i},\ i=0,\ 1,\ 2,\ \ldots \tag{14}$$

$$\Delta_{s_{k,i}}=f_{\mathrm{d}}(\Delta_s, Noise(k,t_i)),\ i=0,1,2,\ldots \tag{15}$$

Here, $Noise(k,t)$ is a 2D fBm noise function. The section of this function for each $k$ is 1D fBm noise function, which has correlation to one another. The function $f_{\mathrm{d}}$ rounds up or down the value of $\Delta s \cdot Noise(k,t_i)$ to a discrete value $\Delta s_{k,i}$ so as to make $s_{k,i}$ ($i$=0, 1, 2, …, $n$) sampling time at which data exists in $D_2$. This mechanism appropriately changes the sampling interval $\Delta s_{k,i}$ for each line $k$ and each frame time $t_i$, and results in realizing natural 3D 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 appropriate sampling interval in Eq.(11), the function is given the condition that $0.75 \le Noise(k,t) \le 1.25$.

## SURFACE RECONSTRUCTION AND RENDERING OF THE WATERBODY

In this section, we focus on the problem of the surface generation from particles. It is difficult to obtain a smooth surface out of particles (Foster and Fedkiw, 2001). Level set method is used in many approaches of particle-based water simulation (Enright *et al*., 2002; Foster and Fedkiw, 2001). In this paper, an alternative method is proposed. We obtain the surface mesh directly from the particles by tiling triangle strips from the outlines of 2D slices.

The tiling process is divided into two steps: extracting the outlines of the 2D simulation; constructing the smooth surface from the 2D outlines.

**Outline extraction of the 2D simulation**

To extract the outlines of the 2D MPS simulation result, we construct an image corresponding to the positions of the particles. In the image, the pixels with foreground color represent the particles. A slice of 2D MPS of particles is shown directly using OpenGL as Fig.5.

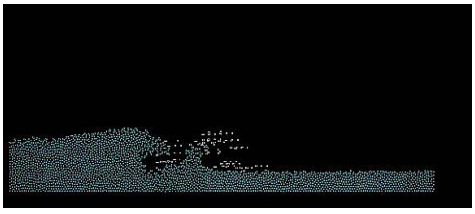


**Fig.5 A slice of 2D simulation**

In the initial state of the MPS computation, the space is divided into a grid. The resolution of the image is the same as that of the grid. The image is filled initially with background color.

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

**Fig.6 Image of 2D simulation of Fig.5**

Then we can extract the outline using some technique of image processing. Fig.6 shows the outline of the water body. The outline is composed of successive pixels in a fixed direction, say anti-clockwise order. Moreover, the extracted outline is not a closed contour. We just take the part of the contour on the top face of the water surface, neglecting the parts on the side faces and the bottom face.

**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.

1. Tiling triangular strip

After the outlines are extracted from 2D slices of particles, we then construct the triangular surface segment by segment.

Given two outline segments $O_1=\{\boldsymbol{P}_i, i=1, \ldots, M\}$ and $O_2=\{\boldsymbol{Q}_i, i=1, \ldots, N\}$, and a weight function $w(\boldsymbol{V}_1,\boldsymbol{V}_2)$, where $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are two successive points on an outline, we tile a triangle strip between $O_1$ and $O_2$ (Fig.7). We define $W(\boldsymbol{P}_i,\boldsymbol{P}_{i+k})=\sum_{j=i}^{i+k-1} w(\boldsymbol{P}_j,\boldsymbol{P}_{j+1})$. In the initial step, we set $i=j=1$.

There are many tiling criteria (Christiansen and Sederberg, 1978; Ekoule *et al*., 1991; Meyers and Skinner, 1992). Here we choose the following criterion.

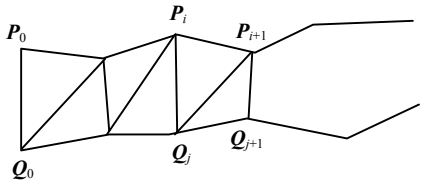



**Fig.7 Tiling of triangles between two outlines**

If Eq.(16) is satisfied, then the triangle $\boldsymbol{P}_i\boldsymbol{P}_{i+1}\boldsymbol{Q}_j$ is added, otherwise the triangle $\boldsymbol{P}_i\boldsymbol{Q}_j\boldsymbol{Q}_{j+1}$ is added.

$$\begin{aligned}&|W(\boldsymbol{P}_0,\boldsymbol{P}_i)+w(\boldsymbol{P}_i,\boldsymbol{P}_{i+1})-W(\boldsymbol{Q}_0,\boldsymbol{Q}_i)|\\&<|W(\boldsymbol{Q}_0,\boldsymbol{Q}_i)+w(\boldsymbol{Q}_j,\boldsymbol{Q}_{j+1})-W(\boldsymbol{P}_0,\boldsymbol{P}_i)|.\end{aligned}\tag{16}$$

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

$$w(\boldsymbol{V}_1,\boldsymbol{V}_2)=|\boldsymbol{V}_1\boldsymbol{V}_2|/L,$$

where $L$ is the total length of the outline segment, and $|\boldsymbol{V}_1\boldsymbol{V}_2|$ is the length of line segment $\boldsymbol{V}_1\boldsymbol{V}_2$.

But this function cannot deal with concave outlines, as it may produce distorted surface from concave contours. In the following section we describe the algorithm for the calculation of the weight function.

2. Calculation of the weight function

For the non-convex case, we have to deal with the concave section on the outline. The method of decomposition and projection is adopted to calculate the weights. Given a contour $C=C^0=\{\boldsymbol{P}_i, i=1, \ldots, M\}$, $H^0$ is the convex hull of $C^0$. $\boldsymbol{P}_{i_1}$ and $\boldsymbol{P}_{j_1}$ are 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 $\boldsymbol{P}_{i_1}$ and $\boldsymbol{P}_{j_1}$, and this set is defined to be 1-order concave section. In Fig.8, we have $C^1_{(3,10)}=\{\boldsymbol{P}_3, \boldsymbol{P}_4,\ldots, \boldsymbol{P}_{10}\}$.

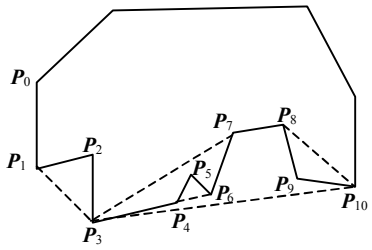



**Fig.8 Decomposition of a non-convex contour**

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

We project a point $\boldsymbol{P}_i$, $i\in(i_n, j_n)$, to a point $\boldsymbol{P}_i'$ on the line segment $\boldsymbol{P}_{i_n}\boldsymbol{P}_{j_n}$, as shown in Fig.9. Its coordinate $(x_i', y_i')$ is calculated as follows:

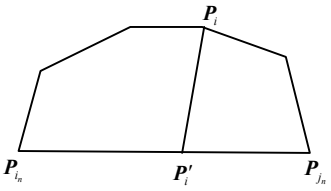



**Fig.9 Projection for concave section**

$$\begin{cases} x_i' = x_i + R_i(x_{j_n} - x_{i_n}), \\ y_i' = y_i + R_i(y_{j_n} - y_{i_n}), \end{cases} \tag{17}$$

where $R_i$ is given as:

$$R_i = \frac{\sum_{k=i_n}^{i-1} d(\boldsymbol{P}_k, \boldsymbol{P}_{k+1})}{\sum_{k=i_n}^{j_n-1} d(\boldsymbol{P}_k, \boldsymbol{P}_{k+1})}. \tag{18}$$

The above process can be continued recursively, until all the vertices are projected onto the initial convex-hull $H_0$.

Assuming that $\boldsymbol{P}_i$ is finally projected to $\overline{\boldsymbol{P}}_i$ on $H^0$, we calculate the weight of $\boldsymbol{P}_i\boldsymbol{P}_{i+1}$ as follows:

$$w(\boldsymbol{P}_i, \boldsymbol{P}_{i+1}) = |\overline{\boldsymbol{P}}_i\overline{\boldsymbol{P}}_{i+1}|. \tag{19}$$

After the weights of the two outlines have been calculated, the tiling process continues according to the tiling criterion Eq.(16).

**Smoothing the reconstructed surface**

To decrease particle simulation noise, denoising operations are implemented on the extracted outlines. The denoising operations perform in intra outline and between adjacent outlines.

The first phase of smoothing is done after the outline is computed. We set the coordinates of the vertex as the weighted sum of the coordinates of itself and the proceeding and succeeding vertices on the same outline.

The second phase 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 coordinates of this vertex as the weighted sum of the coordinates of the vertices in the neighborhood.

## RENDERING OF THE SPLASH EFFECT

Splashing effect is an important feature of a breaking waves scene. Currently, little is known about the rendering of the splashing effect. Takahashi *et al*.(2003) have recently proposed a splash and foam representation. 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 2D simulation is considered 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 certain lifespan. After being displayed during their 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. The techniques below are used to avoid such visual artifacts:

(1) The threshold value is changed randomly for each line;

(2) The number of new splash particles is changed according to the speed of the original particle;

(3) The lifespan is selected randomly.

In order to efficiently generate long period animation using fewer simulation steps, a data segment containing a few typical wave pushing and washing up motions, short interval $<s_a, s_b>$ in a 2D 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 Eq.(14) given by accumulating the sampling intervals $\Delta s_{k,i}$ of Eq.(15) 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 Fig.10, $\Delta k$ was set to 20.

## RESULTS

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

We have conducted experiments on two datasets. The experiment on dataset 1 shows breaking wave washing against a wharf, and the experiment on dataset 2 shows the wave washing upon a beach.

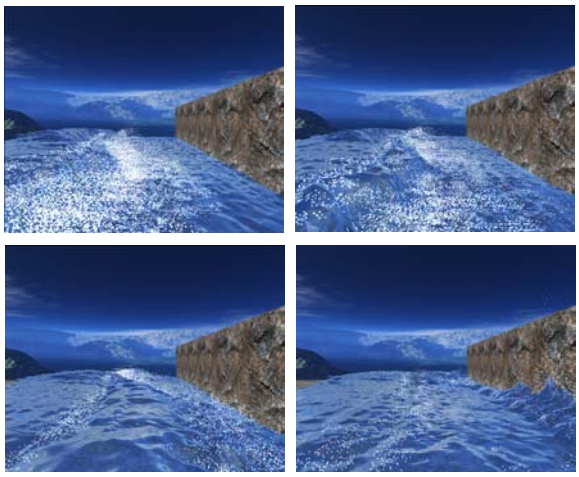
Fig.10 shows several frames of the breaking wave animation. The images show the water surface of the waterbody, splash effects and environment mapping.

Table 2 shows some parameters in the experiments. The performance is measured on a computer with 2.0 GHz Pentium IV CPU, 512 MB memory and NVIDIA GeForce FX 5700 display card.

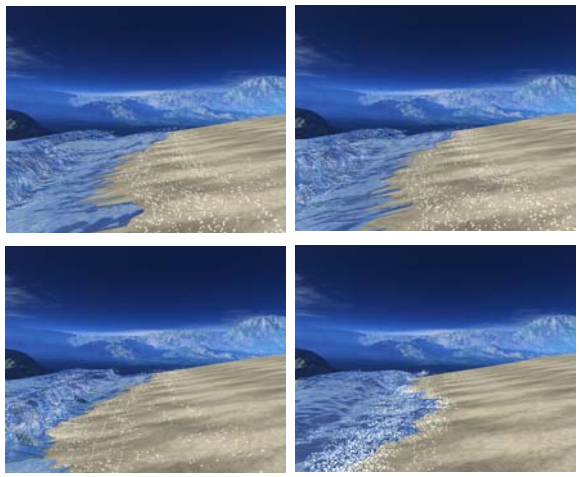
**Table 2 Summary of experiments**

| Parameter | Beach | Wharf |
|---|---|---|
| Total number of particles | 2903×256=766208 | 3993×256=1022208 |
| Resolution | 256×128×256 | 256×128×256 |
| Average rendering time per frame (s) | 0.31 | 0.45 |

Our rendering efficiency is competitive compared with the previous method of particle-based simulation (Foster and Fedkiw, 2001; Premoze *et al*., 2003). In (Premoze *et al*., 2003), it takes 3 min to generate a frame in the scale of 100000 particles, while 7 min was required to render a frame in (Foster and Fedkiw, 2001). The proposed method can render more than 2 frames per second in an equivalent scale of millions of particles. Experimental results showed the efficiency of our method in surface generation out of particles.



(a)



(b)

**Fig.10 Frames of the breaking wave animation. (a) Waves washes against a wharf; (b) Waves washes upon a beach**

## CONCLUSION AND FUTURE WORK

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

Our experiments showed that the method proposed in this paper generates very efficiently the waterbody surface, and that it is easy to expand to large scale using this method.

In the experiments, we used fixed number of particles. In future, we may explore the method to generate breaking waves using "infinite" particles.

Our experiments are based on 2D MPS. We would like to apply our surface generation method when fast 3D MPS solver is available.

ACKNOWLEDGEMENT

The authors would like to express thanks to Prof. Koshizuka, the University of Tokyo, for his source code of 2D MPS, and Mr. Toru Suzuki, for the source code of particle-based simulation of breaking wave.

**References**

Christiansen, H.N., Sederberg, T.W., 1978. Conversion of complex contours line definition into polygonal element mosaics. *Computer Graphics*, **12**(2):187-192.

Ekoule, A.B., Peyrin, F.C., Odet, C.L., 1991. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, **10**(2):182-199. [doi:10.1145/108360.108363]

Enright, D., Marschner, S., Fedkiw, R., 2002. Animation and Rendering of Complex Water Surfaces. Proc. ACM SIGGRAPH 2002. San Antonio, Texas, USA, p.736-744.

Foster, N., Fedkiw, R., 2001. Practical Animation of Liquids. Proc. ACM SIGGRAPH 2001. Los Angeles, California, USA, p.23-30.

Fournier, A., Reeves, T., 1986. A simple model of ocean waves. *ACM Transactions on Graphics*, **20**:75-84.

Fujimoto, T., Miyauchi, S., Suzuki, T., Chiba, N., 2005. Noise-based Animation of Waving Phenomena. IWAIT2005. Jeju, Korea, p.459-464.

Jeschke, S., Birkholz, H., Schmann, H., 2003. A Procedural Model for Interactive Animation of Breaking Ocean Waves. Proc. WSCG2003 POSTERS.

Koshizuka, S., Tamako, H., Oka, Y., 1995. A particle method for incompressible viscous flow with fluid fragmentation. *Comput. Fluid Dynamics*, **4**:29-46.

Mandelbrot, B.B., 1977. The Fractal Geometry of Nature. W.H. Freeman, New York, p.127-135.

Meyers, D., Skinner, S., 1992. Surfaces from contours. *ACM Transactions on Graphics*, **11**(3):228-258. [doi:10.1145/130881.131213]

Mihalef, V., Metaxas, D., Sussman, M., 2004. Animation and Control of Breaking Waves. Proc. ACM SIGGRAPH/ Eurographics Symposium on Computer Animation. Grenoble, France.

Peachey, D.R., 1986. Modeling waves and surf. *ACM Transactions on Graphics*, **20**:65-74.

Premoze, S., Tasdizen, T., Bigler, J., Aaron L., Whitaker, R., 2003. Particle Based Simulation of Fluids. Proc. Eurographics 2003. Granada, Spain, p.401-410.

Song, O.Y., Shin, H., Ko, H.S., 2005. Stable but nondissipative water. *ACM Transaction on Graphics*, **24**(1):81-97. [doi:10.1145/1037957.1037962]

Stam, J., 1999. Stable Fluids. Proc. ACM SIGGRAPH'99, p.121-128.

Stam, J., Fiume, E., 1995. Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes. Proc. ACM SIGGRAPH'95, p.129-136.

Takahashi, T., Fujii, H., Kunimatsu, A., Hiwada, K., Saito, T., Tanaka, K., Ueki, H., 2003. Realistic Animation of Fluid with Splash and Foam. Proc. Eurographics 2003. Granada, Spain, p.391-400.