Real-time Dynamic Simulation of 3D Cloud for Marine Search and Rescue Simulator

Helong Shen[1], Yong Yin[1], Yongjin Li[1,2] and Pengcheng Wang[3]

[1] *Key Laboratory of Marine Simulation and Control, Dalian Maritime University*
[2] *Vessel Command Department, Public Security Marine Police Academy*
[3] *Laboratory of Naval Architecture and Ocean Engineering, Dalian Maritime University*

***Abstract*— As the main scenery of sky, the effect of 3D cloud influences the fidelity of visual system and the immersion effect of simulator. In this paper, based on the work of Y. Dobashi and T. Nishita, small region Cellular Automaton is generated and more realistic cloud simulation is improved. The experimental results show that the visual system of simulator can run in real-time and has a relatively higher refresh rate after changeable 3D cloud being applied.**

***Index Terms*—Cellular automaton, Dynamic cloud simulation, Wind simulation, Marine search and rescue simulator.**

I. INTRODUCTION

Marine Simulator, whose future trend is Marine Search and Rescue Simulator (shorted for MSRS) that can be used for integrative maritime search and rescue training, is a kind of Man- In-the-Loop-Simulations system[1]. The fidelity of visual scene, which is the important component of MSRS, has a great effect to the immersion of whole system. When people is trained in the simulator, especially in the Search and Rescue Helicopter simulating unit in MSRS, the realistic display of changeful cloudscape is required. According to this, how to draw a realistic cloud dynamically to perfect the reality of the visual system and improve the immersion effect & depth cue of MSRS has a great significance. This paper studies on the dynamic simulation of 3D cloud and makes an improvement to Nishita-Dobashi's method. An interactive system is implemented to create dynamic 3D cloud which can be used for the visual system of MSRS in real-time.

II. PREVIOUS WORK

There are two categories of cloud simulation in recent years: static and dynamic cloud simulation. The former one is mainly based on texture mapping on geometries without any physical model simulation. After the sky box is built, different kinds of cloud texture are mapped to display the visual scene.[2-16] have advanced or improved many kinds of 3D cloud simulation. The dynamic 3D cloud simulation in this paper is based on the work of T. Nishita and Y. Dobashi, who have improved cloud simulation using Cellular Automaton (shorted for CA).

The cloud simulation in Marine Simulation right now mainly uses 2D texture mapping technology realized by *OPENGVS API* or VEGA API, etc.. The advantage of this kind simulation is that the design and implementation is simple. The only thing user has to do is to map the cloud texture on the sky box. Because the texture is real photo of different kind of clouds, the display of cloud is accords with the actual sky scene under the same weather condition. This method has advantages such as the computation time is short, memory cost is very small and the visual system can run in real-time. That's just what the MSRS requests. However, this method also has disadvantages. The 2D cloud created doesn't have hiberarchy. But if observing carefully, the real clouds in daily life is far lower than sky and different cloud has different altitude. Real cloud has growth, extinction, advection by wind and various shape and many other characters. To simulate these characters, the physical model of cloud is needed, which can't be realized by 2D cloud. So the sky display in the marine simulator is not so realistic. But the realistic scene display is just emphasized many times in the Specification of Visual System in the Standard for Certification of Full Mission Shiphandling Simulator (Class A) brought forward by Det Norske Veritas (DNV)[1, 17].

III. MODELING OF DYNAMIC 3D CLOUD

3.1 The Growth Simulation

The CA model was advanced first by Nagel and used to simulate static 3D cloud[18]. The physical simulation process of the cloud growth is described as follows. The simulation space is compartmentalized into the cellular group with cells number of $nx \times ny \times nz$, and for simplicity the cells are aligned parallel to the *xyz* axes. For every cell, three variables, *hum*, *act*, *cld*, are set. Each represents vapor, phase transition factor and clouds. These variables are Boolean types, whose value is 0 or 1. *Hum* =1 means that there is enough vapor to form clouds, *act* = 1 means the phase transition from vapor to water (clouds) is ready to occur, and *cld*=1 means there are clouds. The transition rules of these variables at each cloud cells are depicted as follows.

Manuscript received on April 15, 2008.
E-mail: shenhelong@126.com

$$hum(i,j,k,t+1) = hum(i,j,k,t) \wedge \neg act(i,j,k,t)$$
$$cld(i,j,k,t+1) = cld(i,j,k) \vee \neg act(i,j,k,t) \quad (1)$$
$$act(i,j,k,t+1) = \neg act(i,j,k,t) \wedge hum(i,j,k,t) \wedge f_{act}(i,j,k)$$

Where f_{act} is a Boolean function and its value is calculated by the status of act around the cell. The following function is used by taking into account the fact that cloud grow upward and horizontally.

$$f_{act}(i,j,k) = act(i+1,j,k,t) \vee act(i,j+1,k,t)$$
$$\vee act(i,j,k+1,t) \vee act(i-1,j,k,t) \vee act(i,j-1,k,t)$$
$$\vee act(i,j,k-1,t) \vee act(i-2,j,k,t) \vee act(i+2,j,k,t) \quad (2)$$
$$\vee act(i,j-2,k,t) \vee act(i,j+2,k,t) \vee act(i,j,k-2,t)$$

For more details about the rules please refers to [13]. In this paper, during the simulating process, users can set different initial values for *hum*, *act*, *cld* according to the actual cloud conditions such as cloud area, cloud thickness, and the growth speed of cloud and so on. Under the rules mentioned above, cloud growth can be simulated well and cloud can be shaped into variable kinds. The simulating effect of cumulus is better.

3.2 Cloud Extinction

The CA model mentioned above has a disadvantage that is at one cell the variable *cld* would never change after becoming to 1. So the cloud extinction never occurred at this cell. Dobashi uses the method bellowed to deal with the problem[2]. In order to simulate cloud extinction, one threshold p_{ext} is set, to specify cloud extinction probability. At each cloud whose *cld* is 1, a random number *rnd* $(0 \le rnd \le 1)$ is generated and *cld* is changed to 0 if $rnd < p_{ext}$. By changing the probability at each cell at different times, the cloud extinction can be realized. But there still is a problem; clouds are never generated after extinction. Based on the same principle, phase transition probability and vapor probability p_{act}, p_{hum} are set randomly to change the variable *act*, *hum* to 1 randomly. The methods described in this section are summarized by the following transition rules.

$$cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \wedge IS(rnd > p_{ext}(i,j,k,t_i)),$$
$$hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \vee IS(rnd > p_{hum}(i,j,k,t_i)), \quad (3)$$
$$act(i,j,k,t_{i+1}) = act(i,j,k,t_i) \vee IS(rnd > p_{act}(i,j,k,t_i)),$$

Where *rnd* is an uniform random number, and *IS(e)* is a Boolean function that returns 1 if expression *e* is true, otherwise returns 0.

IV. CA MODEL IMPROVEMENT AND DYNAMIC CLOUD

4.1 Implementation of CA Model

In this paper, we use the method mentioned above to generate 3D cloud. However, after integrated into the visual scene system of MSRS that is a kind of system which is running under the common PC in real time, the cloud couldn't take from the refresh rates of the whole scene. So to generate cloud like Dobashi's method is not so suitable. Because Dobashi says in[2], if the model space is 256×128×20, the time to generate a cloud picture with the size of 640×480 is about 10s. However, Specification of Visual System (Class A) prescribe that the scene generating speed is no less than 30 fps. Considering other sceneries loaded in the scene system such as ships, coast constructions, wharfs, sea, lighthouses and so on, the 3D cloud of Dobashi's is unusable. So we make some changes to Dobashi's CA model. We specify a small region to build a separate CA model space for generating each piece of needful cloud. Every model evolves separately. The redundant calculation at non-cloud area is spared. So the computation reduces greatly

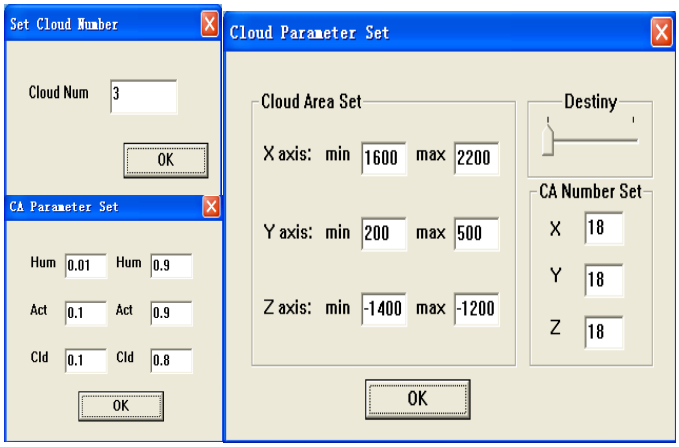

Fig. 1. The parameters set dialog

In this paper, aiming at simulating dynamic 3D cloud, an interactive system is set up. Users can specify the number of cloud pieces due to different purpose. As shown in Fig.1, we can see the cloud piece-number set dialog, CA parameters set dialog including some parameters mentioned above, the relevant cloud parameters set dialog including the cloud generating area and so on. If the piece-number is more than 1, the last two dialogs will create several times for each piece of cloud. Using these dialogs, users can initialize different cloud with different parameters easily, such as cloud altitude, display area, growth and extinction speed and so on. Changeable clouds can be set.

4.2 Shape Control of 3D Cloud

The CA model mentioned in Section 4.1 for each piece of cloud, which cost less computation, still remains another problem. We can find from the introduction above that the cells area specified by users is like a cuboid. So the shape of the cloud generated toward this method has singleness and was unreal. That is not what we want to do. In order to shape original clouds differently, we use an ellipsoid to control it. According to our knowledge of solid geometry, each cuboid has an ellipsoid whose size and figure are close, inscribed ellipsoid for example. In practice, we base on the cloud area set by users to find a relevant ellipsoid and generate its equation. The general ellipsoid equation is described as follows.

$$\frac{(x-\Delta x)^2}{a^2} + \frac{(y-\Delta y)^2}{b^2} + \frac{(z-\Delta z)^2}{c^2} = 1 \quad (4)$$

We can control these ellipsoid parameters *a*, *b*, *c*, Δx, Δy, Δz to make the ellipsoid changeable. Beside the arbitrary CA model region, the joint of the cuboid and the ellipsoid of specified cloud can have different original shapes. The following figures (Fig.2) show two of these conditions.

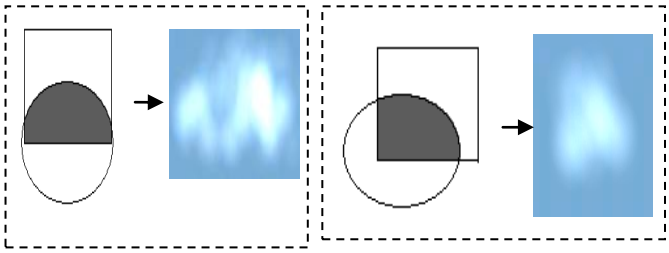

Fig. 2. Shape control using ellipsoid

4.3Continuous Density Distribution Calculation

The real cloudscape in our daily life has a continuous density distribution. But the density obtained from the simulation has only two values, which is 0 and 1. So a smoothing method should be used to generate realistic cloud. We use the method in[4]. The density calculation equation is given as follows.

$$\rho(x,t_i)=\sum_{i,j,k\in\Omega(x,R)}^{N} q(i,j,k,t_i)\,f(|x-x_{i,j,k}|) \tag{5}$$

Where N is the number of cells in the spheriform area $\Omega(x,R)$ with the radius *R*. *q(i, j, k, t_i)* is the density of the metaball placed at each cell at time t_i, which is given by the following function.

$$q(i,j,k,t_i)=\frac{1}{N_c}\times\sum_{l.m.n\in\Omega(x_{l.m.n})}^{n_c} cld(x_{l.m.n}) \tag{6}$$

Where *cld* is the binary distribution and $x_{l,m,n}$ is the coordinate of the grid *(l, m, n)*. In (5), *f* is the field function given as follows.

$$f(r)=\begin{cases}-\frac{4}{9}a^6+\frac{17}{9}a^4-\frac{22}{9}a^2+1 & (r\le R)\\ 0, & (r>R)\end{cases} \tag{7}$$

In this paper, we set the spherical area with the radius of $\sqrt{2}l$ (*l* is the length of each cell grid) as a metaball for each cell. After density calculation, we load textures according the result value. Because we use 2D texture to map the cell, when the view direction is changed, the 2D texture will not face the viewer any more. So we use the billboard method to settle the problem. We set a billboard at each metaball. Then calculate the current view matrix, set the billboard matrix by the view matrix. After matrix translation, the normal direction of the billboard is always up against the view direction. So the viewer can see the frontispiece of billboard at anytime.

4.4 Improvement of Wind Simulation

The wind field simulation used by Y. Dobashi is in single direction with fixed speed[2]. However, the uniform wind field is rare in real world. The velocity and direction are different depending on the height from the ground. But if simulating the wind physically, we have to do many calculation of aerodynamics. As a result the computation will increase too much to make the system run in unreal-time. We deal with the problem in this way. In actual cloud simulation, the wind $\bar{v}$ was split into v_x, v_y, v_z, which is parallel to the *xyz* axes respectively. The velocity along the axis is different if the wind has different space vector. By this method, the wind moves in a more realistic way with little calculation increased. So we can get more changeable cloudscape display running in real-time. The translation rules are given as follows.

$$hum(i,j,k,t_{i+1})=\begin{cases}hum(i-v_x,j-v_y,k-v_z,t),\\ \quad i-v_x>0,j-v_y>0,k-v_z>0\\ 0,\ otherwise\end{cases}$$

$$cld(i,j,k,t_{i+1})=\begin{cases}cld(i-v_x,j-v_y,k-v_z,t),\\ \quad i-v_x>0,j-v_y>0,k-v_z>0\\ 0,\ otherwise\end{cases} \tag{8}$$

$$act(i,j,k,t_{i+1})=\begin{cases}act(i-v_x,j-v_y,k-v_z,t),\\ \quad i-v_x>0,j-v_y>0,k-v_z>0\\ 0,\ otherwise\end{cases}$$

Where v_x, v_y, v_z are calculated as follows and shown in Fig.3.

$$\begin{cases}v_x=|\vec{v}|\times\cos\alpha\times\cos\beta\\ v_y=|\vec{v}|\times\cos\theta\times\cos\gamma\\ v_z=|\vec{v}|\times\cos\alpha\times s\end{cases} \tag{9}$$

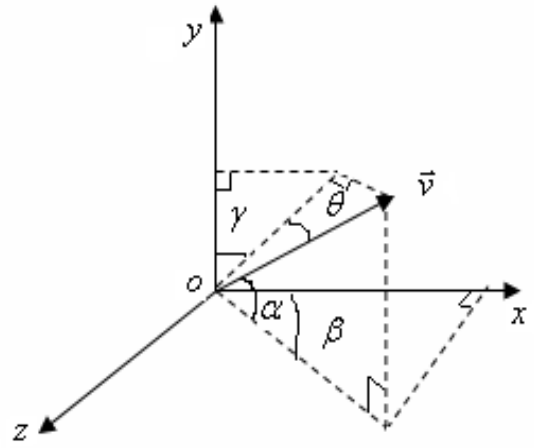

Fig. 3. Wind calculation

4.5 The Flow Chart of Cloud Simulation

In this section, we introduce the flow chart of cloud simulation in practice. First, users initialize the correlative parameters for clouds with the set dialogs in section 4.1, including the number of cloud pieces, CA parameters, clouds' parameters and so on. Then the program initializes every piece of cloud according to the input values, such as initializing CA models, setting ellipsoid for each model to control original shape, loading all clouds texture photos etc. For next, we active

the CA growth and extinction rules, set a metaball for each cell grid, calculate its density, calculate the wind velocity and direction along each axis if winding. Followed above steps, we translate the current matrix, set billboard for each metaball. At the end we select a cloud texture depending on the density for each billboard, map the texture, add lighting effect and end cloud rendering. If other sceneries have been rendered, we begin next frame rendering.

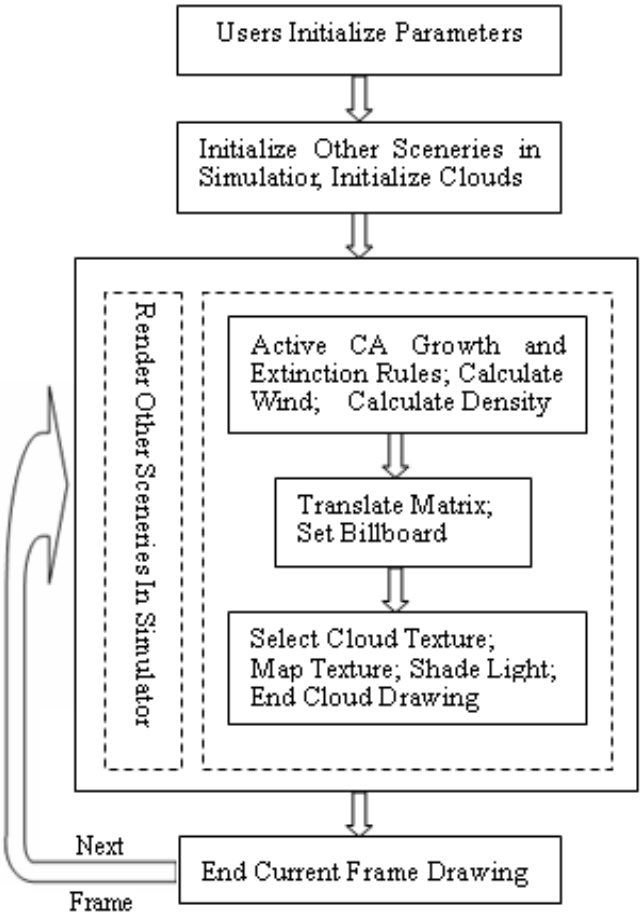

Fig. 4. Cloud simulation flow chart

V. RESULTS AND ANALYSIS

In this paper, based on Visual C++ 6.0, using OpenGL API, which is accelerated by hardware, we build an interactive system to set CA models to simulate dynamic 3D clouds and display it in the Visual System of MSRS. The computer configuration we used here is shown in Tab.1. After loading many models into the navigating scene including wharfs, vessels, coast constructions, mountains, sea, sky scene etc., we got the test result shown in TABLE. 1 when cloud is generated using our CA models. This table shows that, the refresh rate of the whole system is more than 35 fps when the system runs on a computer with relative low-lying capability. The 3D cloud generated can be used in visual scene of MSRS. In Fig.5, *(a)* and *(b)* show three piece cloud generated with the CA model of 28×28×18 at time 2s and 8s, where is no wind blowing. *(c)* shows the two pieces of cloud generated with the CA model of 35×35×25 after system running 1 second, the wind velocity is 2*m/s* with direction 80° in *xoy* plane. *(d)* and *(e)* show the wind simulation. Blown by the wind of 4*m/s*, with the direction like unit vector *(1, -1, 1)*, 6*s* later, *(d)* changed to *(e)*.

VI. CONCLUSION

In this paper, we generated dynamic 3D clouds which can be used in MSRS Visual Scene based on the Dobashi's CA model. Because we have to ensure the refresh rate to fulfill the whole system running in real-time, CA model and wind simulation are

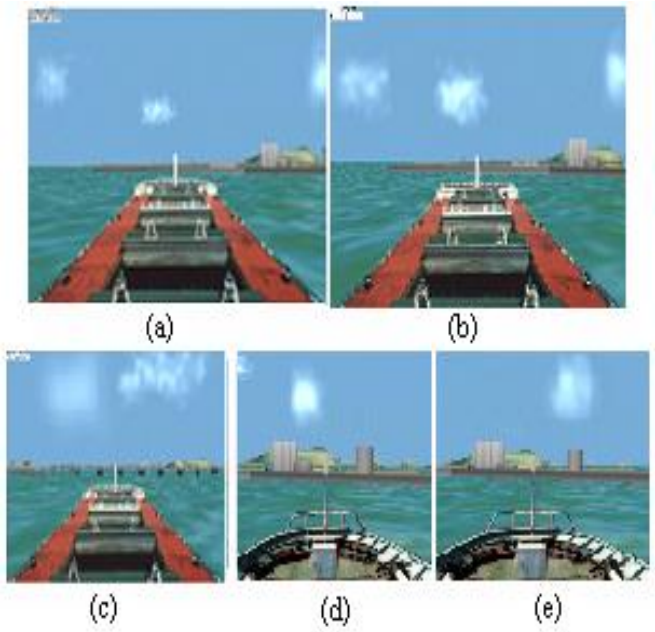

Fig. 5. Cloud test pictures

improved. An interactive system which is realized by OpenGL API is built to generate changeable clouds according to users' need. The experimental results show that the visual system of simulator can run in real-time and has a relatively higher refresh rate after 3D cloud being integrated. There still are some points we have to perfect. The lighting model is too simple. In this paper, fixed light source is set at infinite place. We should consider more about refraction and scattering when light passing through the cloud. Because of the complexity of MSRS, we need to improve the refresh rate higher. The future work we want to do is to translate the rendering process of clouds into CG language, which is GPU based and can improve the render speed and quality greatly.

TABLE. 1. THE RESULTS OF CLOUD TEST

Model cells Number	Cloud Pieces	Resolution of Computer	Refresh Rate(fps)	Computer Configuration
28×28×18	3	1024×768	40~50	P4 2.0 GHz Nvidia Geforce FX 5200 DDR 512M O/S Windows XP
28×28×18	3	800×600	50~55	
35×35×25	2	1024×768	35~40	
35×35×25	2	800×600	40~50	

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their helpful suggestions and comments, and also thank the support by the 973 Program of China (No. 2002CB312103).

REFERENCE

[1] Y. Jin, Y. Yin, H. Ren *et al.*, Visual System in Marine Simulator *Journal of Dalian Maritime University*, vol. 27, no. 2, pp. 16-21, 2001.

[2] Y. Dobashi, K. Kaneda, H. Yamashita *et al.*, A Simple, Efficient Method for Realistic Animation of clouds. *SIGGRAPH 2000*, pp. 19-28, 2000.

[3] Y. Dobashi, K. Kusumoto, T. Nishita *et al.*, Feedback Control of Cumuliform Cloud Formation based on Computational Fluid Dynamics. SIGGRAPH2008, pp. Article 94, 2008.

[4] Y. Dobashi, T. Nishita, and T. Okita, Animation of Clouds Using Cellular Automaton. CGIM'99, pp. 251-256 1999.

[5] D. S. Ebert, Volumetric Modeling with Implicit Functions: A Cloud is Born. SIGGRAPH'97, pp. 147, 1997.

[6] D. S. Ebert, and R. E. Parent, Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-Buffer Techniques. SIGGRAPH'90, pp. 357-366, 1990.

[7] P. Elinas, and W. Stuerzlinger, Real-time Rendering of 3D clouds, *The Journal of Graphics Tools*, vol. 5, no. 4, pp. 34-45, 2000.

[8] G. Y. Gardner, Visual Simulation of Clouds, *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 297-303, 1985.

[9] M. J. Harris, Real-Time Cloud Simulation and Rendering, The University of North Carolina at Chapel Hill, 2003.

[10] M. J. Harris, W. V. B. III, T. Scheuermann *et al.*, Simulation of Cloud Dynamics on Graphics Hardware, *Graphics Hardware*, pp. 92-103, 2003.

[11] M. J. Harris, and A. Lastra, Real-Time Cloud Rendering. Eurographics 2001, pp. 76-84, 2001.

[12] R. Mizuno, Y. Dobashi, and T. Nishita, Modeling of Volcanic Clouds using CML, *Journal of Information Science and Engineering*, vol. 20, pp. 219-232, 2004.

[13] F. Neyret, Qualitative Simulation of Convective Clouds Formation and Evolution. EGCAS 97, pp. 113-124, 1997.

[14] D. Overby, Z. Melek, and J. Keyser, Interactive Physically-Based Cloud Simulation. 10th Pacific Conference on Computer Graphics and Applications (PG'02), pp. 469, 2002.

[15] J. Stam, and E.Fiume, Dipicting Fire and Other Gaseous Phenomena Using Diffusion Processes. SIGGRAPH 95, pp. 129-136, 1995.

[16] J. Stam, and E. Fiume, Turbulent Wind Fields for Gaseous Phenomena. SIGGRAPH 93, pp. 369-376, 1993.

[17] Y. Yong, Research on Real Time Algorithm of Viewing Scene in Distributed Navigational Simulation System, Dalian Maritime University, 2001.

[18] K. Nagel, and E. Raschke, Self-organizing Criticality in Cloud Formation, *Physica A*, pp. 519-531, 1992.

Helong Shen was born in Shandong, China in 1984. He reveived the B.S. from Dalian Maritime University in 2005, and began successive postgraduate and doctoral programs of study in the same year. Now, he is a Ph.D. student in Key Laboratory of Marine Simulation and Control of Dalian Maritime University. His main research interest is real-time marine simulation.

Yong Yin was born in 1969. He received his B.S. and M.S. degree in navigation technology from Dalian Maritime University in 1991 and 1994 respectively. In 2001, he received Ph.D in Transportation Information Engineering and Control from Dalian Maritime University. His research interests include simulation technology, virtual reality and computer graphics.

Yongjin Li was born in Luoyang, China in 1977. He received the B.S. from Dalian Naval Academy, in 1999, and received the M.S. from Shanghai Maritime University in 2004. Now, he is a Ph.D. student in Key Laboratory of Marine Simulation and Control of Dalian Maritime University.

He has been a teacher of Vessel Command Department, Public Security Marine Police Academy, Ningbo, China since 1999. His main research interest is real-time marine simulation, now in realistic sea wave simulation.

Pengcheng Wang was born in Tianjin, China in 1983. He received the B.S. from Dalian Maritime University in 2007, and began successive postgraduate of study in the same year. Now, he is a M.SC. student in Laboratory of Naval Architecture and Ocean Engineering of Dalian Maritime University.