International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:5, No:8, 2011

Real-time Interactive Ocean Wave Simulation using Multithread

K. Prachumrak and T. Kanchanapornchai

***Abstract*—**This research simulates one of the natural phenomena, the ocean wave. Our goal is to be able to simulate the ocean wave at real-time rate with the water surface interacting with objects. The wave in this research is calm and smooth caused by the force of the wind above the ocean surface. In order to make the simulation of the wave real-time, the implementation of the GPU and the multithreading techniques are used here. Based on the fact that the new generation CPUs, for personal computers, have multi cores, they are useful for the multithread. This technique utilizes more than one core at a time. This simulation is programmed by C language with OpenGL. To make the simulation of the wave look more realistic, we applied an OpenGL technique called cube mapping (environmental mapping) to make water surface reflective and more realistic.

***Keywords*—**Interactive wave, ocean wave, wind effect, multithread

I. Introduction

ONE of the most complicated natural phenomena is the fluid's behavior. An ocean wave is one of the examples. There are many levels of the ocean waves; the smallest one is the tiny bubbles on the surface, the medium level is the wave on the surface of the deep ocean and the large one is the breaking of the shallow water wave. [1]

Researches on the simulation of waves can be categorized into three groups. The first one is based on physical models[2], usually from the fluid kinematics equations. Navier-stroke [3] and Fast Fourier Transformation [4] are examples of these models. Because of the complex equations, this kind of wave represents the real physical phenomena but takes time to generate.

The second one is based on geometrical shape. These types of wave surfaces are generated by sine and quadratic equations [5]. This method is fast to present but the waves are less realistic.

The third one is based on oceanography spectrums using frequency spectrum and directional spectrum to create the ocean wave. Although this method applies complex mathematics models, it is faster to generate the wave than the first one. However, the wave from this method looks more realistic and it is more suitable for simulating the deep ocean water. [6]

In this research, the oceanography spectrum technique is adopted and the multithreading technique is applied together with many of the OpenGL rendering techniques for the beautiful and real-time display of the deep ocean wave.

K. Prachumrak and T. Kanchanapornchai are with the Department of Computer Science, King Mongkut's Institute of Technology, Ladkranbang. Bangkok, Thailand.

II. The Oceanography Spectrum Model

A. Wave model

The spectrum model is implemented to analyze the wave by the random process. There have been many researches on the oceanography spectrum. The famous model of this type is Longuet-Higgins model [7][8]. The model is constructed by the wave from many directions, the same as the real ocean wave. This model applies many superimpositions equation as follows:

$$\eta(x,y,t)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_{ij}\ \cos(k_i(x\cos(\theta_p+\theta_j)+y\sin(\theta_p+\theta_j))-\omega_i t-\varepsilon_{ij}), \quad (1)$$

where (x,y) is a point coordinate on the wave surface, t is time vector and $\eta(x,y,t)$ is the height at the point. a_{ij} ω_i k_i represent the amplitude, circle frequency and wave numbers respectively. ε_{ij} is the initial phase of the composed wave, which can be selected randomly between $0 - 2\pi$. θ_p represents the wind direction, θ_j is the deflection angle between the wind and the wave direction.

This model is applied in this research. The following sub-sections describe how to calculate each parameter in terms of the ocean spectrum to simulate the ocean wave.

B. Amplitude a_{ij}

a_{ij} is the amplitude. It can be calculated from:

$$a_{ij}=\sqrt{2S(\omega_i)G(\omega_i,\theta_i)\Delta\omega\Delta\theta}\ . \quad (2)$$

To calculate (2), we need to know that $S(\omega)$ is the frequency spectrum function:

$$s(\omega)=\frac{\alpha g^2}{\omega^5}\exp\left[-\beta\left(\frac{g}{U_\omega}\right)^4\right], \quad (3)$$

and $G(\omega_i\theta)$ is the directional spectrum:

$$G(\omega,\theta)=\frac{8}{3\pi}\cos^4(\theta)\,. \quad (4)$$

C. Frequency ω

The minimum and maximum frequency, ω_{min} , ω_{max}, should be determined first (fig.1). The frequency can be calculated from

$$\varpi_{i=}\varpi_{min}+(i-1)\Delta\varpi+\frac{\Delta\varpi}{2}\qquad(1\le i\le n)\,, \quad (5)$$

where $\Delta\omega$ denotes the frequency interval:

$$\Delta\omega = \frac{\omega_{max} - \omega_{max}}{n}, \quad (6)$$

and n is the number of segmentation in the frequency range from fig.1.

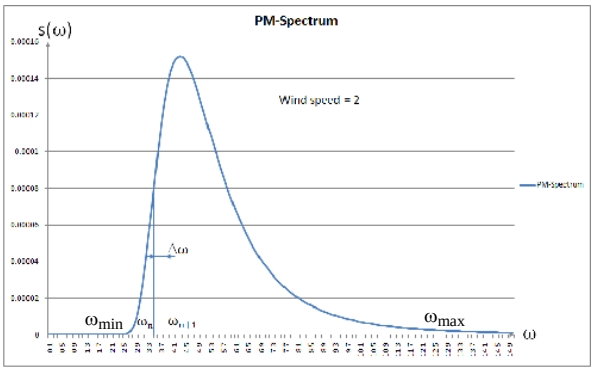

Fig. 1 Energy distribution of spectrum

D. Wave numbers, k

Our research ignored the ocean depth, so the wave number has a relationship with the frequency as follows:

$$k = \omega^2 / g, \quad (7)$$

where g is the acceleration of gravity.

E. Directional angle, θ

The directional angle causes a motion of the ocean wave. Δθ represents the directional angel with the interval.

$$\Delta\theta = \pi / m, \quad (8)$$

and the deflection angle between the wave and the wind direction is:

$$\theta_j = -\frac{\pi}{2} + \Delta\theta(j-1) \quad (1 \le j \le m). \quad (9)$$

According to [8], to create several types of the deep ocean waves, the wave parameter values are recommended as follows: the wind speed, U, varies from 0.55 ~ 20; the frequency ranges, ω_i , are from 0.2 ~ 6; and the frequency interval, Δω, is at 0.1, 0.2 and 0.4.

III. Adding the Wind Model

From (3), at $S(\omega)_1$ the frequency range, the wind direction and the wind speed are $\omega_{1\,min}$ - $\omega_{1\,max}$,θ_{1p} and U_1, respectively. At $S(\omega)_2$, define $\omega_{2\,min}$ - $\omega_{2\,max}$,θ_{2p} and U_2 instead. These parameters are used when the simulation time is at the point t_1-t_2. The effects of the wind can be calculated from the frequency range ω_{min} - ω_{max} as follows:

$$\omega_{max} = \omega_{1max} - \frac{f(t)}{U_2 - U_1}\left|\omega_{2max} - \omega_{1max}\right|. \quad (10)$$

$$\omega_{min} = \omega_{1min} - \frac{f(t)}{U_2 - U_1}\left|\omega_{1min} - \omega_{2min}\right|. \quad (11)$$

Replace (9) and (10) to (6) to get Δω

$$\theta_p = \theta_{1p} + \frac{t}{t_3 - t_2}(\theta_{2p} - \theta_{1p}). \quad (12)$$

IV. Implementation with Multithread and GPU

From the wave model explained above, this research suggests the method to create the ocean wave from the following processes:

1) Define the parameters, allocate the memory and define the function for each thread.

2) Apply the OpenGL technique, **Vertex Buffer Object**. It uses the GPU to display instead of CPU. This technique helps to storage the vertex data to the fast memory directly for the display. We first create a buffer object to store the vertex data. Then we allocate the memory in the video memory for the buffer. Finally, we upload the data to the buffer

3) Apply **Cube mapping** technique, from OpenGL, to create the sky cube texture for the reflection of the water surface. Instead of Ray tracing technique, this Cube mapping technique offers rapid shading with the comparable result.

4) Define the number of vertices used to create the wave, in this research, the vertices = 101x101 =10201, and with the OpenGL function, GL_VERTEX_ARRAY, define the beginning position of each vertex.

5) Define the values of the parameters, time and direction of the wind. Then, calculate the wave model explained previously. This calculation took time, so we applied the multithreading technique to this step.

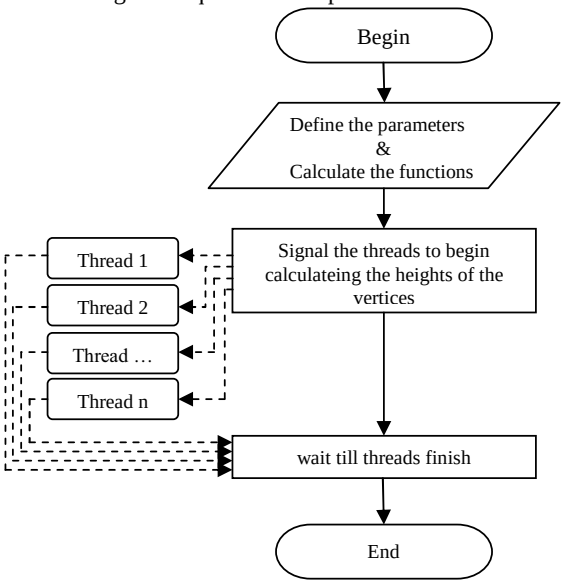

Fig. 2 Flowchart of the multithreading

Another flowchart (fig.2) shows how the threads work simultaneously.

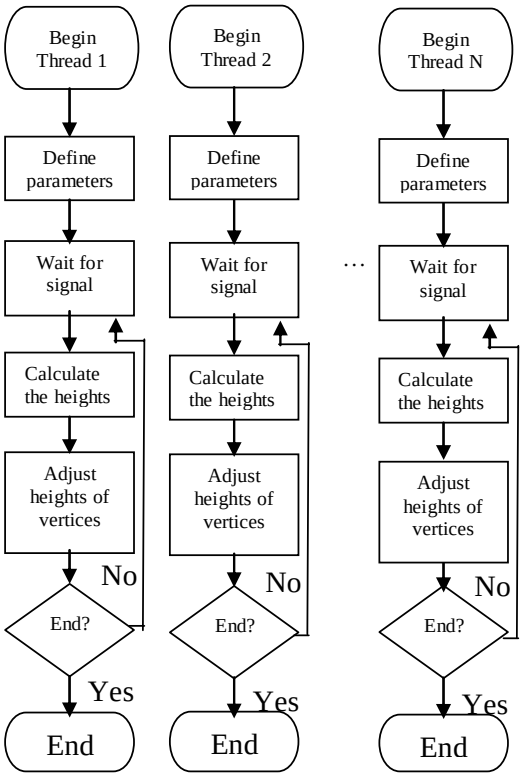

Fig. 3 Flowchart of multithreading technique to calculate height

6) After gaining the height, use *Vertex Buffer Object* technique to draw the surface.

7) For the rendering process, calculate Normal vector of each surface and each vertex. Then, apply Phong model for the shading. As this process was also time-consuming, we applied multithreading technique for faster processing, as shown in fig. 3.

After all these processes, we would be able to simulate the real-time interactive ocean wave.

V. Experimental Results

This research runs on a normal PC. The test bed is:

CPU : Intel Core i7 720QM (1.6 – 2.8 GHz, 8 Cores)
GPU : NVIDIA GeForce GTX 260M
Memory : DDR3 (1066 MHz) 4GB

This research created the ocean wave with 100 x 100 grids, which equals to 101x101 vertices or 10,201 vertices. To simulate the ocean wave, the height of each vertex has to be calculated, and thus it is very time-consuming to create one frame of wave on the screen. With the OpenGL techniques, Vertex array, Vertex Buffer Object Cube mapping and multithreading technique, we can render the wave in real-time. The results are shown in Table I.

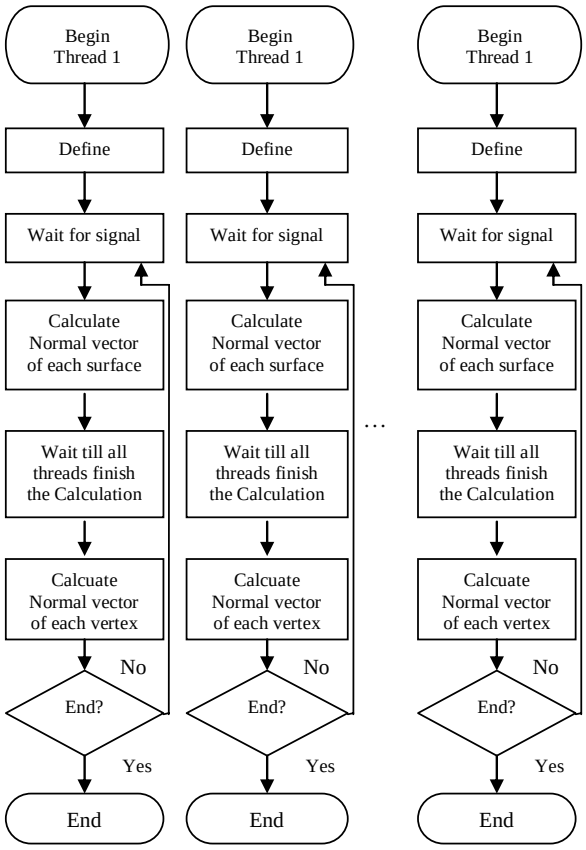

Fig. 4 Flowchart of multithreading technique to calculate Normal vector for the rendering process

Table I shows that this multithreading technique works well on the calculation of the vertices heights but not on the normal map. This is because of the number of the heights is much higher that that of the normal vectors. For the heights the higher the number of threads helping to calculate the height map, the faster the processing time. Compared to the calculation on the height of the 10,201 vertices, the normal map does not need as many calculations. The results indicate that the processing times when applying the multithreading technique to calculate the normal map are not different from those when not applying it, as shown in table 1. When the multithreading technique is applied, the OS and the CPU have to control the thread switching, which slows the processing time.

VI. Summary

The multithreading technique is very useful for the large amount of calculations. It can be applied to accelerate the processing of the CPU which has multi-cores. This research implements the multithreading technique to make the real-time rendering of the interactive ocean wave. This simulation applies the cosine wave superimposition model, which requires the huge amount of calculation of the height of all the vertices

to simulate the wave, so it is really suitable for the multithreading technique.

The OpenGL techniques, Vertex array and Vertex Buffer Object are applied to speed up the processing time. The OpenGL Cube mapping technique is applied to make the reflection and refraction of the surface to make the scene more realistic while consuming a little of the CPU processing power. The experimental results show that the wave from this proposed method looks realistic enough as shown in fig. 5 and real-time as the shown results in Table1. Our wave is good for applications such as games. To show that the wave is interactive, this research adds some objects, cube boxes, floating on the water as shown in fig.6. We apply Archimedes' Principle of Buoyancy to simulate the floating objects on our wave.

Fig. 5 The wave with the frequency 9 and 9 degree, frame rate 56

Fig. 6 The wave interact with some objects

REFERENCES

[1] N. Thurey, M. Muller-Fischer, S. Schirm, M Gross, *IEEE 15th Pacific Conference on Computer Graphics and Applications*, 2007. pp. 39-46.

[2] C. Yuksel, D. H. House, J. Keyser, " Wave particles," *ACM Transaction on Graphics*, Vol. 26, No. 3, Article 99, July 2007, pp.

[3] V. Mihalef, D. Metaxas, M. Sussman, "Animation and control of breaking waves," *ACM SIGGRAPH Symposium on Computer Animation*, 2004, pp. 315-324.

[4] J. Tessendorf, "Simulating ocean water," in Simulating Nature: *Realistic and Interactive Techniques*, SIGGRAPH 2001, Course Notes 47.

[5] X. Ma, Z. Chen, G. Shi, "Real-Time ocean wave motion simulation based on statistic model and GPU programming," IEEE *2nd International Conference on Information Science and Engineering* (ICISE), 2010, pp. 3876 – 3880.

[6] C. Wang, Z. Wang, J. Jin and Q. Peng, "Real-time simulation of ocean wave based on cellular automata," Submit to CAD/Graphics'2003 October, Macao, China.

[7] H. He, H. Liu, F. Zeng, G Yang, "A way to real-time ocean wave simulation," IEEE *International Conference on Computer Graphics, Imaging and Vision: New Trends*, 2005. pp. 409-415.

[8] X. Zhao, F.Li, S. Zhan and Z. Li, "Ocean wave simulation under wind change effect," in *Proc. of the IEEE First International Conference on Innovative Computing, Information and Control*, 2006. pp. 26-29

TABLE I
THE COMPARISON OF THE NUMBER OF THREADS AND THE FRAME-RATE

No. of threads used to calculate Height map	No. of threads used to calculate Normal map	Usage of CPU (%)	Frame per second
1	1	11 - 13	24
2	1	16 - 21	32
3	1	28 - 36	42
4	1	36 - 45	50
5	1	43 - 52	60
6	1	48 – 58	65
7	1	51 – 70	72
8	1	42 - 64	63
1	2	12 - 13	24
1	3	12 – 14	24
1	4	12 – 15	25
1	5	13 – 15	25
1	6	13 – 15	25
1	7	13 – 15	25
1	8	11 – 14	22
2	2	18 – 22	33
4	4	40 – 49	56
7	7	62 – 78	81