Proceedings of the 32nd Chinese Control Conference
July 26-28, 2013, Xi'an, China

Ocean Wave Real-time Simulation Based on Adaptive Fusion

WANG Shunli[1,2], KANG Fengju[1,2] , WANG Dinghua[1,2]

1. School of Marine Technology, Northwestern Polytechnical University, Xi'an 710072, China
E-mail: shunli@qq.com

2. National Key Laboratory of Underwater Information Process and Control, Xi'an 710072, China
E-mail: kangfengju@sina.com

Abstract: Aim at the poor fidelity in large-scale ocean wave scene real-time simulation, a simulation method based on adaptive fusion is proposed. Perlin noise is used to generate height map at distance and FFT method is adopted at near sight. The adaptive fusion of height map is realized during transition phase. Meanwhile, simulation software is designed and the ocean wave real-time simulation with a 100x100km2 range is realized. The method is compared with the ocean wave that only uses Perlin noise method or FFT method. Simulation results show that the method can effectively improve the fidelity of ocean wave simulation in real time.

Key Words: Ocean wave simulation, height map, adaptive fusion

1 Introduction

Large-scale ocean battlefield environment simulation is one of the important contents of battle visual simulation. Its fidelity directly affects the battle system process observation, battlefield situation analysis and fighting effectiveness evaluation. While the performance of real-time simulation can affect operating and training in variously complex mission.

In recent years, many international scientists make related research on ocean wave simulation. [1] performed water face simulation based on the Navier-Stokes equations. Since the method needs a great number of complex calculations, it can not meet the requirement of large-scale ocean wave real-time simulation. Perlin noise was used to generate wave-height map in [2]. It can realize dynamic ocean wave simulation, but lack of details at near sight. [3] generated height map with stacking and repeating multiple gray scale image created by artists, and realized realistic rendering of ocean wave by using the technology based on GPU vertex shader and grid model of concentric circles. [4] simulated ocean wave based on height map that generated by using FFT method, the method can be used in high quality of ocean video rendering or in ocean wave real-time simulation flexibly. [5] used LOD (Level of Detail) method in rendering of ocean surface mesh based on [3], and added reflection and refraction. The method improved ocean wave reality, but in large-scale ocean scene，the wave at distant was too regular and the reality was too poor.

Our ocean wave model is based on Perlin noise and FFT method. Both the advantages of two methods are combined. Perlin noise is used to generate height map at distant, and FFT method at near sight. During the transition phase, the height map is dealt with adaptive fusion algorithm. Meanwhile, we design simulation software which realizes ocean wave real-time simulation with a 100x100km2 range. Comparing with the ocean wave that only uses Perlin noise or FFT method, our approach can effectively improve the fidelity of ocean wave simulation in real time.

*This work is supported by the Ship Pre-research Support Technology Foundation, China (11J4.1.1).

2 Ocean Wave Model

2.1 Perlin Noise Method

Perlin noise-function was first proposed by Ken Perlin in 1985. Essentially, Perlin noise-function is a random number generator, but it is different from general ones. There needs an integer as parameter to generate Perlin noise. The noise can be accessed multiple times with the same value as the result. A fractal noise can be created by layering multiple noise-functions at different frequencies and amplitudes. By using the two-dimensional noise we can generate a random texture. It can be seen how multiple octaves of Perlin noise sums up a fractal noise in Figure 1.

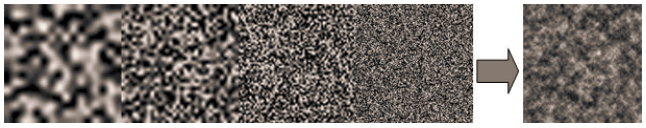

Fig. 1: Multiple octaves of Perlin noise sums up to a fractal noise （α=0.5）

$$fnoise(x) = \sum_{i=0}^{octaves-1} \alpha^i \cdot noise(2^i \cdot x) \qquad (1)$$

Formula 1 demonstrates how a fractal noise can be created from multiple octaves of Perlin noise and how all the amplitudes can be represented by a single parameter (α). The higher values for α, the rougher noise will be looking.

2.2 The FFT Method Based on Phillips Spectrum

The FFT method based on Phillips spectrum first get a series of double frequency sine waves which amplitude and phase is relevant, and then gets height of every grid point on ocean face by calculating with FFT.

Assuming that there are $M \times N$ discrete points on the XZ plane, the vertical offset of the discrete point $x = (nL_x / N, mL_z / M)$ at time t is

$$h(\mathbf{x}, t) = \sum_{\mathbf{k}} h(\mathbf{k}, t) e^{i\mathbf{k}\cdot\mathbf{x}} \qquad (2)$$

where $\mathbf{k}=(k_x,k_y)$, $k_x=2\pi n/L_x$, $k_z=2\pi n/L_z$, $-N/2\le n\le N/2$, $-M/2\le m\le M/2$, $h(\mathbf{k},t)$ is the height amplitude Fourier and can calculate with the Phillips spectrum

$$P_h(\mathbf{k})=A\frac{1}{k^4}e^{\frac{-1}{(kL)^2}}|\hat{\mathbf{k}}\cdot\hat{\omega}|^2 \quad (3)$$

The height amplitude Fourier at time $t=0$ is

$$h_0(\mathbf{k})=\frac{1}{\sqrt{2}}(\xi_r+i\xi_i)\sqrt{P_h(\mathbf{k})} \quad (4)$$

where ξ_r and ξ_i are Gaussian random numbers that meet mathematical expectation 0 and variance 1.

The height amplitude Fourier at time t is

$$h(\mathbf{k},t)=h_0(\mathbf{k})e^{i\omega(k)t}+h_0^*(-\mathbf{k})e^{-i\omega(k)t} \quad (5)$$

where $\omega(k)=\sqrt{gk}$.

Actually, in real motion of ocean wave, the continuous change of wind direction leads to roll wave effect, extrusion between waves make the top of wave sharp and flat at the bottom. The height map generated by our method is the result of stacking of multiple sine waves. The top of wave is too flat and not real, so we need to add a roll wave model. The model can be realized by changing the position of grid in horizontal direction. Here we use the amplitude of ocean wave to get the offset of discrete point X in horizontal direction

$$D(\mathbf{x},t)=-\sum_{\mathbf{k}}(i\frac{\mathbf{k}}{k})h(\mathbf{k},t)e^{i\mathbf{k}\cdot\mathbf{x}} \quad (6)$$

But this method need one more FFT, we simplify it for accelerated computing

$$D_{x,y}=\frac{S}{N}(M_{x+1,y}-M_{x,y},M_{x,\ y+1}-M_{x,y}) \quad (7)$$

where S is the size of FFT grid, N is the number of FFT grid, M is the wave-height map.

3 Height Map Adaptive Fusion

3.1 Adaptive Fusion Algorithm

In large-scale ocean wave scene simulation, Perlin noise method can generate height map quickly, but the fidelity of the simulation result is too low at near viewpoint. The FFT method is fast, but the result is too regular and unreal. We take both advantages of the two methods; use FFT method near the viewpoint and Perlin noise at distant. In order to avoid mutation during transition phase, we propose an adaptive fusion based on a nonlinear function

$$f(d)=\frac{1-e^{-(\frac{d-d_1}{d_2-d})^2}}{1+e^{-(\frac{d-d_1}{d_2-d})^2}} \quad (8)$$

where d is the distance from current grid to viewpoint, $d_1<d_2$.

Observe formula 8, there are

$$\lim_{d\to d_1}f(d)=\lim_{d\to d_1}\frac{1-e^{-(\frac{d-d_1}{d_2-d})^2}}{1+e^{-(\frac{d-d_1}{d_2-d})^2}}=0 \quad (9)$$

$$\lim_{d\to d_2}f(d)=\lim_{d\to d_2}\frac{1-e^{-(\frac{d-d_1}{d_2-d})^2}}{1+e^{-(\frac{d-d_1}{d_2-d})^2}}=1 \quad (10)$$

It can be seen that the function value tend to 0 when $d\to d_1$, and 1 when $d\to d_2$. The adaptive fusion algorithm is obtained through weighting the height h_p computed by Perlin noise method and h_f by FFT method

$$h(x,y)=f(d)h_p(x,y)+(1-f(d))h_f(x,y) \quad (11)$$

where $h(x,y)$ is final height of the ocean wave grid point (x,y).

From formula 9, 10 and 11, we can know that the fusion function value tend to h_f when $d\to d_1$, and h_p when $d\to d_2$. This fusion algorithm not only makes the transition between two height map smoother, but also retain their characteristics. Thus the fidelity of ocean wave simulation is much higher.

3.2 The Realization of Height Map Adaptive Fusion

The height map fusion can be realized in CPU, but it costs a lot of time on computing, as every grid point needs to be judged and fusion. This will seriously affect real-time performance. In current, the technologies of vertex shader and fragment shader are supported by general new graphic processor. So we can read value from height map and fuse the height by programming with the programmable hardware shading language, and then get the height of ocean surface grid point by using texture mapping technology.

Before mapping, we need to compute the grid points and their associated texture coordinates, and then look for the two heights h_f and h_p in the height map according to the coordinates. The final height can be got after processing the heights with adaptive fusion algorithm.

There needs only one computation as the operation of points is parallel in GPU. So the load of CPU will be decreased in real-time simulation.

4 Design of Ocean wave Simulation Software

For testing and verifying the method of this paper, the grid model based on cull and LOD is used. The acceleration technology based on GPU is adopted. The ocean wave simulation software is designed and developed with Visual C++ and OpenGL programming language. The design flow chart is shown in Fig. 2.

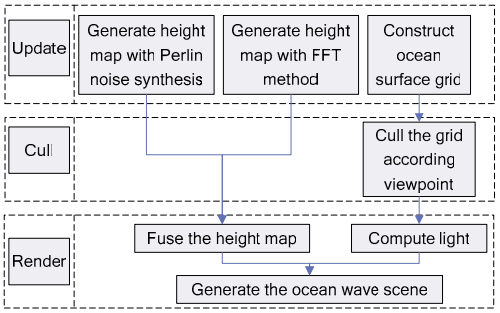

Fig. 2: The design chart of ocean wave simulation software

In update phase, the height map is generated by using Perlin noise and FFT method. The ocean surface grid is constructed. In cull phase, the ocean surface grid is culled according to the viewpoint. At last, in render phase, the height maps are fused with adaptive fusion algorithm in vertex shader, the light is processed in fragment shader, and the ocean wave scene is generated then.

4.1 Grid model based on cull and LOD

Considering the real-time performance of large-scale ocean wave simulation, we use grid model based on cull and LOD. In the grid model, the ocean surface with area of is divided into several small square grids. According to the distance between every grid and viewpoint, we confirm the degree of grid subdivision, namely resolution size, and then finish the LOD of ocean wave grid. The relation of visual range and LOD level is shown in Table 1.

Table 1: The relationship of visual range and LOD level

Visual range	LOD level
0-300	Level 0
300-600	Level 1
600-900	Level 2
…	…

Because the field of view frustum is limited and it can not include the whole ocean surface grid, we need to cull the grid and only generate the grid in the view frustum. The culling algorithm determines whether the grid patch is in the frustum during the dividing of ocean surface grid. If yes, the algorithm needs to decide whether we should divide the grid according the distance between grid patch and viewpoint, or the grid patch should be culled.

4.2 Accelerated processing based on GPU

The Fourier amplitude of every grid needs to be computed in the ocean simulation with FFT algorithm. The grid point of ocean surface is a two-dimensional array and every grid point has the same computation, so that we can apply SIMD (Single Instruction Multiple Data) model to parallel computing in CUDA. The frequency spectrum computation is finished in global static memory. The Fourier amplitude of every grid point is computed in every thread in every thread block. The initial frequency spectrums of all grid points are loaded to the static memory in GPU by using cudaMemcpy() in CUDA API for sharing in all threads. The function of frequency computation is written as device function and is computed in GPU. Every thread reads the frequency spectrum value from static memory and computes formula (5) according the relationship between the thread index and the coordinates of the grid point. After computing the Fourier amplitude, we use CUFFT which is the general mathematics library in CUDA to transform the Fourier amplitude into the height value.

In order to speed up the ocean wave scene rendering, the vertex information and light are processed by programming with the GLSL programmable hardware shading language. We firstly deal with vertex by programming in vertex shader, such as the height-map fusion, the computation of the offset in horizontal direction. The light of ocean surface, reflection and refraction are realized in pixel shader.

5 Experiments and Results Analysis

The experimental environment in this paper is general PC (Core2 Duo E4700 2.8GHz， 2GB Memory， NVIDIA GTX560 Graphics Card). The results of ocean wave scene simulation with different methods can be seen in Figure 3, 4 and 5. Figure 3 is based on Perlin noise method. Figure 4 is based on FFT method. And Figure 5 is the result of adaptive fusion with two methods.

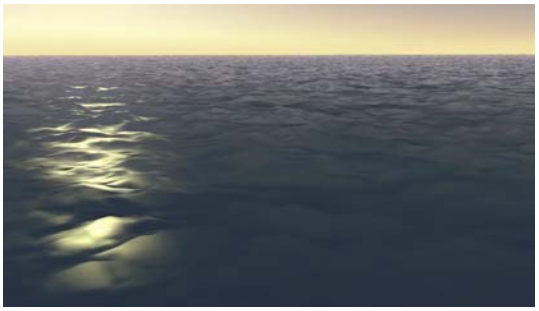

Fig. 3: The result of Perlin noise method

Fig. 4: The result of FFT method

Fig. 5: The result of this paper's method

Table 2: The frame rate results (frames/s)

Resolution	Only Perlin	Only FFT	Perlin and FFT
512x512	551-562	63.08-63.45	61.34-62.06
1024x1024	276-278	20.91-21.89	20.38-20.52

From the table 2 and the figure 3, 4 and 5, we can see that the method of this paper could enhance the fidelity of ocean wave simulation largely and only has a 1.6 f/s decrease in the frame rate.

6 Conclusion and Future Work

The key of this paper is using Perlin noise to generate height map at distant, FFT method at near sight and fuse the height map with adaptive fusion algorithms during the

transition phase. Meanwhile, the simulation software is designed and the ocean wave real-time simulation with a $100x100km^2$ range is realized. This method makes full use of the advantages the high fidelity of Perlin noise at distant and FFT method at near sight. It effectively enhances the fidelity of ocean wave real-time simulation.

In the future, we will research on foam, spray, breaking waves and the wave interacting with entity. Meanwhile we will also start the research on underwater scene simulation for enriching the whole ocean scene simulation.

References

[1] Crane K, Llamas I and Tariq S, Real-time Simulation and Rending of 3D Fluids, in *Nguyen H. GPU Gems 3*. Boston: Addison-Wesley, 2007: 633–675.

[2] Johanson C, Real-time water rendering, *Master thesis in computer graphics*. Sweden: Lund University, 2004.

[3] Kryachko Y, Using Vertex Texture Displacement For Realistic Water Rendering, in *Pharr M. GPU Gems* 2. Boston Addison-Wesley, 2005: 283-294.

[4] Tessendorf J, Simulating ocean water, in *Proceedings of the 29th Annual Conference On Computer Graphics And Interactive Techniques*. USA: New York, 2001: 1-18.

[5] Belyaev V, Real-time simulation of water surface, in *http: // www. graphicon. ru/ 2003/ Proceedings/ Technical/ paper316.pdf*, 2003.