Real-time Simulation of Large Bodies of Ocean Surface Based on Tessellation and Concurrent Kernels

Chen Si
College of Information Science and Engineering
Ocean University of China
Qingdao, Shandong, China
Alex.sichen@gmail.com

Chunyong Ma
College of Information Science and Engineering
Ocean University of China
Qingdao, Shandong, China
chunyongma@ouc.edu.cn

Ge Chen
College of Information Science and Engineering
Ocean University of China
Qingdao, Shandong, China
gechen@ouc.edu.cn

***Abstract*—We present an automatic water simulation method in large scale virtual environment that employed mesh based approaches. To begin with, multi-hierarchy mesh is generated using tessellation technique. Then, CUDA Concurrent Kernels technique is implemented to make Fast Fourier Transform (FFT) and Perlin noise, two ways used to compute sea height map, execute concurrently. The results of these two approaches were combined and used to generate the final sea height. Finally, the illumination effect on sea surface and foam were simulated. This method is successfully applied in VR-GIS(Virtual Reality Geographic Information System) Integrated Three-Dimensional Simulation Platform.**

Keywords-CUDA; Fast Fourier Transform; Perlin Noise; Tessellation; Concurrent kernels

I. INTRODUCTION

With the development of computer hardware and computer graphics, the amount of data processed in the three-dimensional virtual scene rise rapidly. The amount of graphics unit required to draw by each frame increased rapidly. Since the ocean surface is an important natural element in three-dimensional virtual scene, rendering the ocean surface efficiently and realistically becomes an important research topic of computer graphics in recent years.

Build-based simulation method can display the overall characteristics of the water macroscopically, as well as reflect the specific details of the water body to an acceptable level. It is a hot area of large-scale ocean surface rendering in recent years. There are two important issues need to be resolved in build-based method: the mesh generation and the calculation of ocean surface height map. In terms of mesh generation, Yin [1] divided the ocean surface into dynamic surface and static surface; Hinsinger [2] proposed a trapezoidal mesh model; Johanson [3] proposed a projection mesh model to establish a uniform mesh vertical to the sight, and then project it onto the ocean surface of the world space. Computing method of sea surface height map usually use mathematical functions to construct sea wave's shape, which can be divided into two categories: geometric modeling based method and statistical co-spectrum based method. Geometric modeling based method is relatively simple and has easy controlled waveform, it mainly uses trochoid, sinusoidal function and Beta-spline curve to stimulate the geometric shape of waves [4-6]. One disadvantage of this method is that the constructed wave is too regular to be realistic. Statistical co-spectrum based method primarily based on marine statistics and empirical model. Some representative schemes utilize multiple sine function or FFT to synthesis waves that meet the demand of spectral distribution[7-8], or use the method of Perlin noise composition.

In recent years, surface subdivision (Tessellation) technique is paid more and more attention. Tessellation is a technology which decompose polygon into small pieces to enhance geometric fidelity. By means of model mesh segmentation, it makes the model more refined. The development of CUDA platform prompted the emergence of the Concurrent Kernels, which can achieve overlapping execution on two computational streams, take full advantage of GPU resources and improve the efficiency of the parallel computing.

In this paper, Perlin noise and FFT are mixed, realistic rendering of large-scale ocean surface using CUDA general-purpose computing platform and GPU shader is discussed, and a complete ocean surface simulation method is proposed. First we build the ocean surface using level subdivided mesh model based on Tessellation technology; then Perlin noise calculation and FFT calculation are executed concurrently by the CUDA Concurrent Kernels. The two results are mixed to generate wave height chart, then the height of the mesh vertices are gotten using vertex texture displacement technology; at last the ocean surface illumination effect and other special effects are simulated, including ocean surface reflection and refraction of light, Fresnel phenomenon and the foam effect.

II. TESSELLATION SEA GRIDS MODEL

A tessellation self-adapting mesh model, viewpoint related, is represented which segments mesh automatically according to the distance between mesh and viewpoint. Compared with the conventional LOD(Level of detail) mesh model, this self-adapting model has two main characteristics: subdivided mesh points are dynamically generated, without processing; the tessellation procedure in GPU, so the sufficiency is high. To be specific, multi-hierarchy mesh structure is used to simulate sea surface which means that basic mesh is divided into different regions where the degree of tessellation is determined by the distance to the

Published by Atlantis Press, Paris, France.

viewpoint. First, the X-Z plane is identified as the mesh plane, building up a mesh model as basic mesh with equal interval through 3-D modeling software. Second, a perpendicular is made from the position of viewpoint to the mesh plane, defining the intersection as point C and determining a previously segmented square region with point C as its centre. Finally, this square region is divided into some sub-regions, according to the distances between sub-region frontiers and point C. Division levels are determined, referring to the position of C, in other words, the meshes closed to the C are segmented intensively and vice versa. An example illustrating the hierarchy subdivision is shown in Fig.1.

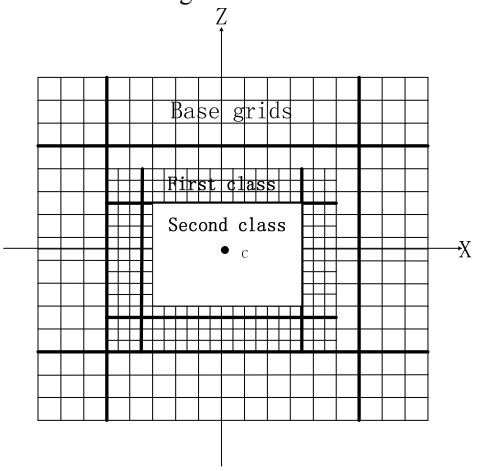

Figure 1. Segmented region is divided into two sub-regions: first class region and second class region. The mesh density is of first region is higher than base mesh, and is lower than that of second region.

In GPU tessellation pipeline, we process the base mesh using Algorithm 1.

Algorithm 1 summarizes our tessellation mesh solver

Algorithm 1 Tessellation of mesh

1: Compute position P_{proj} for viewpoint P_{view}
2: **for** i = 0 up to Number of Quads **do**
3: Compute P_{center} of Q_i
4: LOD_i =N- Distance(P_{proj} - P_{view})/D_{span}
5: Tess = sin((LOD_i * π)/(2*N)) * N
6: OutTess=Tess , InTess=Tess/2
7: GPU Tessellation()
8: **end for**

During each frame, the projection point of the view point is calculated in CPU. Then the projection point is passed into the GPU tessellation pipeline. Every quad on the mesh experiences the tessellation process simultaneously (lines 2 to 8). Specifically, the central position of each quad is computed. We got a distance level between the central position and the projection point. Then we can get the level of LOD (line 4). To ensure the quads near the view point more detailed while those far less, the sinusoidal function has been employed to compute tessellation level (line 5). Before the GPU fixed tessellation pipeline, the out tessellation level and the inner tessellation level can get from the basic tessellation level (lines 6 to 7).

Due to the differences of mesh resolutions in different sub-regions, it is evitable to generate T-style fissure.

III. GENERATING HEIGHT MAP MIXED FFT AND PERLIN

The sea wave is implemented by assigning different height value to every mesh point. These height values are sampled from previously calculated height buffer.

After analyzing present sea height calculating models, we find that Perlin approach has higher efficiency, and that FFT method shows good performance especially in illustrating the near-sighted details. During our experiments, FFT approach based on Phillips Ocean Wave Spectrum demonstrates realistic sea waving and interaction between ocean surface and wind in near-sighted region, but when the virtual scene becomes large enough and far-distance sea waves are focused on, the repeated wave spectrums make the wave simulation lack fidelity. In contrast, except for the inadequate reality in the near-sighted side, the simulation effect is acceptable as a whole, when using Perlin method. In this paper, height map hybrid technique is employed, which is like that: firstly each result of FFT computing and of Perlin computing is abstracted as a height map respectively, and then these two maps are mixed seamlessly by using Hermite interpolation approach.

A. The Theoretical Basis of FFT

Equation (1) is expression of FFT obtained from wave based statistical model and empirical model,

$$h(\vec{p},t) = \sum_{\vec{k}} \tilde{h}(\vec{k},t)\exp(i\vec{k}\bullet\vec{p}) \qquad (1)$$

where $h(\vec{p},t)$ is the height field of sea surface, $\vec{p} = (p_x, p_z)$ represents the position of mesh point, the t means a time point, and $\vec{k}$ is a wave vector.

This function explains a successive process of sea surface height alteration. Since vertexes on the sea surface mesh are discrete, so the values in (1) should be discrete and constrained in a certain way. Assume that the mesh aspect ratio of rendering cycle is $N:M$, then actual length-width distance will be L_x and L_z. So the sea level of mesh discrete vertex in XOZ plane will be $w(|k|) = \sqrt{g|k|}$, where two-dimensional vector $\vec{k} = (k_x, k_z)$ will be expressed by $\vec{k} = (2\pi n / L_x, 2\pi m / L_z)$, n and m are both integer and limited to $-N/2 \le n \le N/2$ and $-M/2 \le m \le M/2$ respectively. $\tilde{h}(k,t)$ is the wave amplitude. When t=0,

$$\tilde{h}_0(\vec{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{p_h(\vec{k})} \qquad (2)$$

And in other general situation

$$\tilde{h}(\vec{k},t) = \tilde{h}_0(\vec{k})\exp(iw(|k|)t) + \tilde{h}_0^*(-\vec{k})\exp(-iw(|\vec{k}|)t) \qquad (3)$$

where $w(|k|) = \sqrt{g|k|}$. Equation (3) shows that height field only depends on the current fourier state, not results at any other time. So this method achieves an efficient way to

calculate elevation map and is appropriate for parallel computing in GPU.

B. The Simulation of Choppy Waves

On the real sea surface, the peak of wave is quite sharp, but the trough is almost smooth. The general method to achieve this result is to increase the height of the wave when it peaks. Instead of changing the height field, mesh vertices can just make horizontal movement to simulate this effect using FFT method. The function below shows the process:

$$\vec{p} = \vec{p} + \lambda D(\vec{p}, t) \tag{4}$$

$D(\vec{p}, t)$ is the horizontal translation amount:

$$D(\vec{p}, t) = \sum_{\vec{k}} -i \frac{\vec{k}}{|\vec{k}|} \tilde{h}(\vec{k}, t) \exp(i\vec{k} \bullet \vec{p}) \tag{5}$$

$\vec{p}$ represents the sea horizon of mesh points, λ is a constant controlling the horizontal displacement. The peak of the wave will become sharper, while the trough is smooth, when the value of λ is increasing gradually. This method can make the wave even more real. It's important to note, however, when the value of is λ oversize, there will emerge overlapping phenomenon between the peaks. Then, it is unable to express the sea height by the height function. This paper adopts Jacobian Determinant to detect the overlapping phenomenon which can measure the extensional ratio of area element from $\vec{p}$ to $\vec{p} + \lambda D(\vec{p}, t)$. Therefore when a vertex has horizontal displacement, its Jacobian Determinant is 1. And if it does not, the Jacobian Determinant would be less than 1. Specially, if the overlapping phenomenon occurs on this dot, the Jacobian Determinant would be less than 0.

C. Detailed Implementation

Hermite interpolation function is used to integrate the merits of FFT and Perlin, which means that in the near-sighted region FFT results play a dominate role in sea height while in the far-sighted region Perlin is the main influencing factor. At the same time, the smooth change from near-sighted regions to far-sighted regions has been guaranteed safely. The interpolation details are showed in

$$Ratio = -2 * (\frac{D_{cur} - D_{\min}}{D_{\max} - D_{\min}})^3 + 3 * (\frac{D_{cur} - D_{\min}}{D_{\max} - D_{\min}})^2 \tag{6}$$

where Dcur is the distance from mesh point to view point, Dmin represents the minimum threshold value and Dmax means the maximum threshold value.

The final sea height for each mesh point is computed in

$$H_{final} = (1 - Ratio) * H_{FFT} + Ratio * H_{Perlin} \tag{7}$$

where HFFT is the height of current mesh point computed by FFT, HPerlin is one computed by Perlin, and Hfinal is the final result mixed these two heights.

GPU Concurrent Kernels are use to execute FFT computing process and Perlin computing process simultaneously in GPU. In the CUDA operating mechanism, a stream is a series of order executive instructions. A stream can be divided into three processes: data transmitting from host to device, kernel commanding and data transmitting from device to host. Since in this paper the objective is to solve the height map computing problems by both FFT and Perlin, actually, the data size is not huge. The upper limit of the memory size is 16 Mega Bytes, even if the resolution of images is 2048*2048. Therefore, the main cost is in kernel computing. Generally, devices which are upper CUDA 2.x support overlap kernels, then CUDA computing resources can be made full use of to improve efficiency.

As the Fig. 2 shows, FFT initial data and Perlin original texture are generate in CPU. There are four streams used to compute Perlin noise height map, Perlin noise normal map, FFT height map and FFT normal map respectively in CUDA where each two of them are concurrent. More specifically, Perlin height map computing stream and FFT height map computing stream are synchronous, while Perlin normal map computing stream and FFT normal map computing stream are synchronous. Finally, the computation data will be transmitted into GPU shader. The sea surface mesh will be generated after the computation by operations such as Tessellation, displacement mapping, etc.

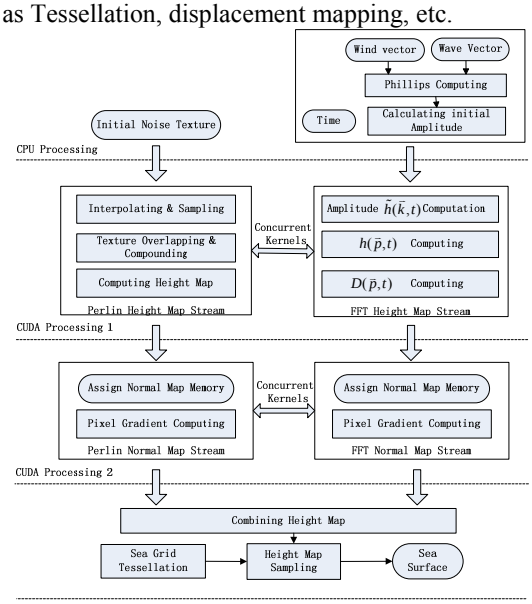

Figure 2. The process of sea surface generating.

Step1: According to the (2), the initial amplitude value can be computed after computing after computing the Phillips frequency spectrum based on the wind vector and the wave vector.

Step2: The original texture which used to conduct Perlin noise computation is generated by transmitting initial high-definition noise image into program.

Step3: CUDA Concurrent Kernels operations are initialized. More specifically, Perlin height texture buffer, FFT height texture buffer and FFT horizontal shift texture buffer are assigned. Also, CUDA kernel computation units are assigned to Perlin noise computation and FFT

Figure 3. Left: reflection Mid: refraction Right: foam

computation respectively. Then, Perlin noise height map stream and FFT height map stream are conducted synchronously through CUDA Concurrent Kernels mechanism.

Step4: Equation (3) is used to compute FFT amplitude value- $\tilde{h}(\vec{k},t)$. Then, horizontal offset texture - $D(\vec{p},t)$, and FFT height texture - $h(\vec{p},t)$ are calculated through (5) and (6) respectively.

Step5: A couple of noise textures are generated by sampling original noise texture in accordance with different step lengths in different hierarchy. Combination of Perlin height map is conducted by overlapping these noise textures in different definitions.

Step6: Perlin noise normal texture buffer and FFT normal texture buffer are assigned. Then, CUDA kernels are distributed to Perlin normal map computation and FFT normal map computation respectively.

Step7: The normal vector in each pixel is created from the gradient vector which is generated through the gradient computation of FFT height map.

Step8: Two vectors, which are along the x axis and along the z axis respectively, for each pixel in Perlin height map, are computed through calculating color differences among the four pixels nearby. After that, every two vectors are crossed to generate Perlin normal map.

Step9: Firstly, according to the distance between view point and each mesh vertex, original mesh is subdivided in GPU Tessellation shader. Additionally, (4) is employed to translate mesh points. The sea surface map is created by smoothly combining Perlin noise height map and FFT height map. Finally, this map is sampled to compute the final sea mesh through displacement map technique.

IV. LIGHT AND SPECIAL EFFECTS

In this paper, cube texture mapping technique, reflection texture blending method and Phong shading model are used to simulate the reflection effects of sky, sceneries above the sea surface and sun respectively. In terms of surface refraction, the submarine depth texture is generated by view clipping and camera migration, and then sea surface refraction effect is created by combining the submarine depth texture and the sea color. Additionally, Fresnel law has been introduced to mix surface reflection and refraction. Finally, according to the result of Choopy calculation we can confirm where and when to generate foam and time-control system has been employed to simulate the rolling and blanking of foam. Results show in the Fig 3.

V. RESULT

The configuration of experimental computer:

CPU Intel Core i5, 2.80GHz,
RAM DDR3,4G
VGA Geforce GTX 450,1G, 1566MHz
OS Windows7

The ultimate purpose of employing CUDA multi-kernels parallel architecture is to improve the operation efficiency of surface rendering. Furthermore, there are two factors affecting efficiency significantly, which are the mesh resolution and the sea surface height computing. So we conducted contrast experiment on height map computing under different resolution ratio. Then we picked 128*128, 512*512, 1024*1024, 2048*2048 resolutions respectively to generate height map by Perlin and FFT.

The comparison experiment processed in CPU and GPU respectively, and we calculated the time each frame costs. As table 1 shows, when mesh resolution is lower than 1024*1024, the speedup of GPU, relative to CPU, is increasing with the mesh resolution getting lager. However, restricted by the quantity of registers in CUDA and parallel unit, the speedup ratio of GPU is on a declining curve.

TABLE I. GPU SPEEDUP RATIOS OF DIFFERENT INITIAL MESH RESOLUTIONS

Initial Mesh Resolution	*CPU(ms)*	*GPU(ms)*	*Speedup ratio*
128*128	31.68	4.01	7.90
512*512	132.23	7.50	17.63
1024*1024	220.78	10.57	20.88
2048*2048	321.56	18.30	17.57

In general, GPU costs less than 20ms each frame, even though the resolution is 2048*2048. The experimental results show that the algorithm proposed in this paper has very great applicable value in large-scale three-dimensional scene. Containing over 1 million triangles, every virtual scene, is rendered at 30 frames per second when the definition of sea surface mesh is 2048*2048.

As the Fig. 4 shows, a large-scale sea surface scene is constructed on VRGIS, a three-dimensional simulation platform, to support the proposed algorithm. It contains lots of three-dimensional scenes, like massive sea waves, wharfs, buildings and terrains.

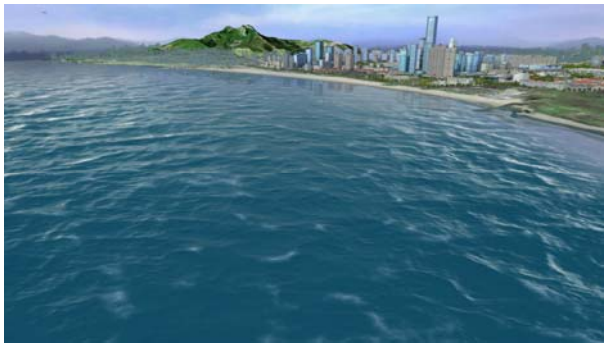

Figure 4. The integration sea surface in virtual scenes

VI. CONCLUSION

This paper presents a view-dependent Tessellation adaptive water surface mesh segmentation algorithm. Based on fixed-size ocean surface mesh, the size of mesh is adjusted by dynamically generate the surface mesh nodes, the amount of computation of mesh sampling is optimized, while the high definition detail is reflected on the sea surface region which is near the viewpoint. Besides, Perlin noise height map generation and FFT generation are executed concurrently by the CUDA concurrent kernels on the CUDA general-purpose computing platform. The highly parallel computing and easy access of graphics card are fully exploited to accelerate ocean surface height map calculation. The efficiency of the entire simulation is improved and the application effects show the method has good application prospect.

REFERENCES

[1] Yin Y;Jin Y C;Ren H X Wave simulation of visual system in marine simulator based on wave Spectrums.

[2] Hinsinger D, Neyret F, Cani M P. “Interactive animation of ocean waves” Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Anication, San Antonio, 2002,pp. 161-166.

[3] Johanson C. Real-time water rendering: introducing the projected gird concept [D]. Lund: Lund University, 2004.

[4] Fournier A, Reeves W T. “A simple model of ocean waves”. ACM SIGGRAPH Computer Graphics, 1986, 20(4), pp. 75-84.

[5] Peachey D R., “Modeling waves and surf,” ACM SIGGRAPH Computer Graphics,1986, 20(4), pp.65-74.

[6] Thon S, Dischler J M, Ghazanfarpour D. “Ocean waves synthesis using a spectrum–based turbulence function” Proc.International Conference on Computer Graphics, Geneva, 2000,pp. 65-72.

[7] Tessendorf J. “Simulating ocean water”Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, Los Angeles,2001,pp. 1-18.

[8] Fréchot J. “Realistic simulation of ocean surface using wave spectra” Proceedings of the 1st International Conference on Computer Graphics Theory and Applications, Setúbal, 2006,pp. 76-83.

Published by Atlantis Press, Paris, France.