GPU-Based Real-time Simulation and Rendering of Unbounded Ocean Surface

Xudong Yang Xuexian Pi Liang Zeng Sikun Li
School of Computer, National University of Defense Technology, Changsha, 410073, P.R.China
yxd336@tom.com

Abstract

We present a multi-resolution mesh model of the ocean surface based on a straightforward terrain LOD scheme, Tiled Quad-tree, with that the region of ocean surface can be extended limitlessly and readily adapted for GPU acceleration. We have introduced the concept of Wrapped Fractal Surface (WFS) for generating height field map of the ocean. Through WFS self-mapping, we can create a series of WFS and obtain the height field map at discrete times without repeating tiles. By interpolating the height map at regular intervals we can build an ocean surface continuously and dynamically. This paper also studied shallow ocean waves concerned about affects of seaboard and underwater terrain. Experimental results showed that our approach was effective and would satisfy the requirements of a realistic real-time navigation for large-scale seascape.

1. Introduction

Real-time simulation and rendering of unbounded ocean surface have attracted more and more attention from researchers because they are widely used in simulation training and 3D games.

Ocean surface is massive, and ocean waves are complex. The movement of ocean waves is determined by various types of forces, such as gravitation, wind speed, wind direction, underwater terrain, and so on. With the process of graphics hardware, the GPU-based algorithms have a better rendering system that can be applied in real-time. This gives us an opportunity for real-time rendering of unbounded ocean surface.

But, in the current papers there is not much information about way of creating multi-resolution mesh for large-scale ocean surface rendering with good quality LOD schemes. This paper offers a possible solution to this challenge.

Our contribution is to present a framework for real-time simulation and rendering of large-scale ocean surface.

Our framework divides the whole large-scale ocean surface into square blocks with different resolution. And with those blocks we create a Tiled Quad-tree of ocean surface. To take full advantage of GPU programmable pipeline, Batched LOD method has been used to enhance the rendering speed. This way, we can make view-culling and LOD choices more conveniently.

This paper use Wrapped Fractal Surface (WFS) for generating height field map coupled with the multi-resolution planar ocean surface mesh. The WFS was built by the famous Perlin noise. The height map of continuous time-point can be generated dynamically in the process of rendering.

The remainder sections of this paper are structured as follows: Section 2 briefly summarizes previous work. Section 3 describes the generation of multi-resolution mesh and LOD of ocean surface. Section 4 explains the height field map using WFS. Section 5 discusses the shallow ocean waves. Section 6 reports results. Section 7 gives some conclusions.

Our rendering pipeline can be divided into several stages as shown in Figure 1.

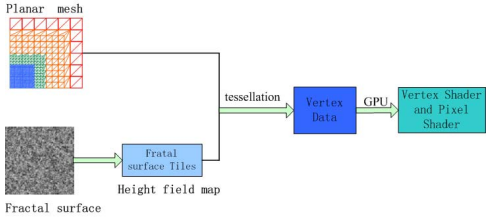


Figure 1. **Rendering pipeline of ocean surface**

2. Previous Work

To our knowledge, presently there are mainly four approaches for modeling ocean surface:

Ninth International Conference on Computer Aided Design and Computer Graphics (CAD/CG 2005)
0-7695-2473-7/05 $20.00 © 2005 **IEEE**

2.1. Based on geometrical models

Peachy [1] adopted the linear combination of sine function and quadratic function to simulate the geometrical shape of waves. Thon [5] used a hybrid approach of trochoids and spectrum only applicable in the calm sea case. Ts'o and Basky [9] represented the ocean waves using beta-splines.

Generally speaking, the above approach is very simple and near real-time, but the scene is less realistic.

2.2. Based on physical models

Ram [10] used simplified numerical method to solve the Navier-Stokes equation for animation of water waves. Joe [6] adopted FFT to simulate the waves. Foster [7] proposed a modified semi-Lagrangian equation to simulate viscous liquids around objects. Wang [19] presented a new approach using cellular automata and put forward a new evolution rule of cell is to mimic the motion of ocean waves.

Those approaches have a good quality of waves, but need a more complex computation that can't simulate large-scale ocean surface in real-time.

2.3. Based on statistical models

Tessendorf [19] showed that dispersive propagation can be managed in the frequency domain and that the resulting field can be modified to yield trochoid waves. Mastin[8] introduced surface wave synthesis that is based on the sum of sinusoidal amplitudes and phases based on empirical observations of oceans. Premoze[3] combined the physical models and oceanography models, but only suit to calm sea.

Since the mathematic model and computation are very complex, those ways are fit to animation and not to real-time.

2.4. Based on time-varying fractals (i.e. Perlin noise)

Perlin [4] used noise synthesis approach to simulate the appearance of the ocean surface seen from a distance.

It can be seen as a particular kind of stochastic fractal that is generated as a summation of several appropriately scaled and dilated copies of a continuous noise function. Johanson [11] just adopted this approach to simulate ocean waves, but only a small area of sea surface.

This way can be implemented very simple and have a more photorealistic ocean. So, in this paper we take Perlin noise to generate the height field map of unbounded ocean surface.

Besides the above four types of ocean waves models, Stefan [16] presented a procedural model for breaking ocean waves. Jensen [17] used different wave model approaches for interactive deep-water animation. Jason [25] presented a GPU-based multi-band Fourier domain approach to synthesizing and rendering deep ocean waves, but he has not considered LOD schemes.

3. Multi-resolution mesh of ocean surface and LOD scheme

The previous methods of ocean surface mesh basically used uniform grid or LOD have not taking into account. Some works consider the LOD, such as [12] and [13], but the LOD is oversimplified. If we want to implement unbounded ocean surface, we must use LOD algorithm to reduce the numbers of triangle patches.

With the development of GPU, the times of reducing triangle patches with Continuous LOD in CPU are so much than the times of rendering patches directly in GPU. Moreover, regular mesh is apt to create display list and vertex array by GPU, with this way we can improve the rendering speed. So in present, the algorithm based on Tiled Quad-tree ([22], [23]) have a newly advantages.

Simultaneously, the Batched LOD has achieved a more applications in terrain rendering. Levenberg [21] presented CABBT. Instead of manipulating triangles, the CABTT algorithm operates on clusters of geometry called aggregate triangles. Since aggregate triangles stay fixed over several frames, they may be cached on the video card. This further reduces CPU load and fully utilizes the hardware accelerated rendering pipeline on modern video cards. Cignoni [20] processed a BDAM technique. BDAM is based on a paired tree structure tiled quad-tree for texture data and a pair of bin-trees of small triangular patches for the geometry. Ulrich [24] had the similar idea. He built an aggregated LOD algorithm called chunked LOD.

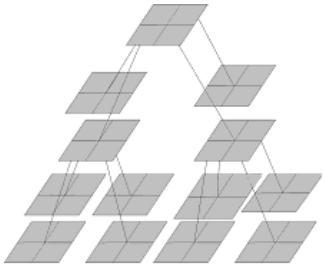


Figure 2. **Tiled Quad-tree**

Thus, we want to combine the advantages of Tiled Quad-tree and Batched LOD for sufficient make use of programming pipeline of GPU. Our algorithms as follows:

This paper divides the whole large-scale ocean surface into a Tiled Pyramid and builds a Tiled Quad-tree, each tile size is $(2^n+1)\times(2^n+1)$. Through constructing the Visible Quad-tree (VQT) within view-frustum and Renderable Quad-tree (RQT) satisfied "remarkability". We can create a multi-resolution LOD tree with VQT and RQT for estimate ocean surface tiles.

Remarkability defines the ratio between screen error and physical space. In short, if we have much lowly node can pass the discriminant of the remarkability, we need choose the ocean tiles with higher LOD.

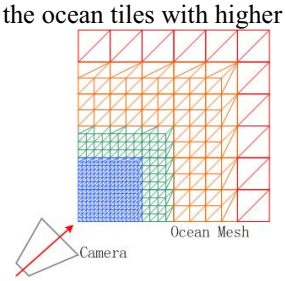


Figure 3. **Ocean surface mesh without view-culling, different resolutions are stitched**

We present the "Joint Index Template" to solve the cracks between tiles with different LOD. This can be implemented simply in GPU. To stitching the tiles seamlessly, the level of two neighbor ocean tiles must not be greater than 1. The quad-tree that satisfied this limited condition is named as "restricted quad-tree". Our mesh of ocean surface is as shown in Figure 4.

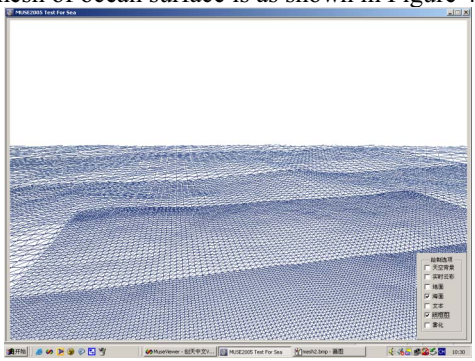


Figure 4. **Ocean surface mesh**

4. WFS and Height field map

In section 3, we create a multi-resolution mesh of planar ocean surface using Tiled Quad-tree. This section mainly produces the height field map using Wrapped Fractal Surface and introduces the tessellation scheme.

4.1. Fractal surface

The sum of multiple octaves of Perlin noise results in a fractal noise. By layering multiple noise-functions at different frequencies and amplitudes, a more interesting Fractal surface can be created. The frequency of each layer is usually the double of the previous layer, which is why the layers usually are referred to as octaves.

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

Equation (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. Higher values for α will give a rougher looking noise whereas lower values will make the noise smoother [11].

We can obtain a Fractal surface as shown in Figure 5.

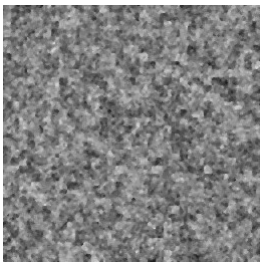


Figure 5. **Fractal surface**

4.2. Mapping Fractal surface to height map

We call the Fractal surface as the Wrapped Fractal Surface (WFS), its size is 512×512 in our experiment, because that we wrap the Fractal surface for expending to 513 × 513. Thus, the boundary of WFS can be stitched seamlessly as shown in Figure 6.

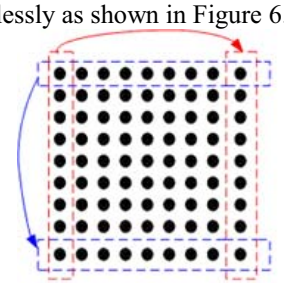


Figure 6. **Wrap the Fractal surface**

While mapping the Fractal surface to height field map, we must define the datum mark as $(offsetX, offsetZ)$. We suppose the resolution of Fractal surface is $n\times n$ and the maximal resolution of fractal tiles is $m\times m$, the interval of samples is l meter. Thus, the coordinate of datum

mark $(offsetX, offsetZ)$ will be some multiples of $m \times l$.

Because the man's view is limited, the ocean region can be seen is also limited. So we map the Fractal surface repeatedly along x and z axes. This way we obtain an unbounded ocean surface. The arbitrary block of ocean surface (x, z) and corresponding fractal tile (i, j) has a relation like equation (2).

$$\begin{aligned} i &= ((x - offsetX)\%(n \times l))/(m \times l) \\ j &= ((z - offsetZ)\%(n \times l))/(m \times l) \end{aligned} \tag{2}$$

4.3. Self-mapping of WFS and constructing a series of WFS

As shown in Figure 7, we take the one or more multiples of lowest tile's size to be the parameter of self-mapping. So we have total 32×32 numbers of shapes (Our WFS is 512×512). We just cache only single WFS in memory, with that we can obtain enough number of Fractal surfaces at each discrete time for generating dynamic ocean surface.

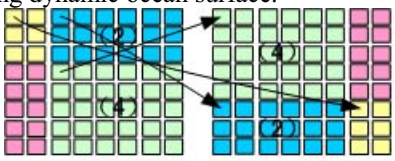


Figure 7. **Self-mapping of WFS**

4.4. WFS Synthesis of discrete times

At time t, if $t_{i-1} \le t \le t_i$, the Fractal surface of ocean height map can got by interpolating of adjoining Fractal surface at t_{i-1}, t_i and t_{i+1} times.

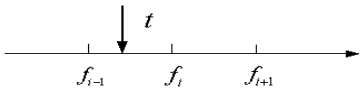


Figure 8. **Interpolation of Fractal surfaces**

For calculating the height data at time t, we have the formula as equation (3):

$$\begin{cases} f_t = w_1 * f_{i-1} + w_2 * f_i + w_3 * f_{i+1} \\ w_1 = \cos\left[\frac{\pi}{3}(\Delta p + 2)\right] \\ w_2 = \cos\left[\frac{\pi}{3}(\Delta p + 1)\right] \\ w_3 = \cos\left[\frac{\pi}{3}(\Delta p)\right] \end{cases} \tag{3}$$

In this equation, we have: $\Delta p = (t - t_{i-1})/(t_i - t_{i-1})$.

As shown in equation (3), the thresholds w_1, w_2 and w_3 are dynamically charged with time.

5. Shallow ocean wave

In section 4 we have been introduced the generation of deep water. Towards the shallow ocean wave, we also make some research. In related works, Layton [14] used a physically based model for animating water waves that is based on the two dimensional shallow water equations. Manuel [15] detailed presented an accurate theoretical model of wave refraction over shallow water.

Shallow ocean wave is a quite complicated problem. Except for the influences of wind speed, wind direction and tide phenomena, the other important affect is seaboard and underwater terrain. The ocean waves near the coast trend to transmit with the direction of vertical coast and result in slowing down and aligning with the coastline. The wind direction will determine the initial direction of the wave rays, which is the path traveled by any given point on the wave, while the wind strength will determine the wave amplitude in deep water. We use a set of sinusoidal to simulate the initial shallow water waves, as described in [15].

We construct a contour line of coast terrain and underwater terrain, with that we can create a shallow water model by using the phase of wave in spreading process.

As shown in Figure 8, shallow ocean waves transmit from *M* to *N*, and vector *MN* is upright to coastline. We compute the difference between the ocean wave phase of *M* and the forecasting phase value of *N*, and can obtain the model of shallow ocean wave concerned with the terrain's influence. Simultaneously, we add the impact of underwater terrain and have a better result, as shown in Figure16.

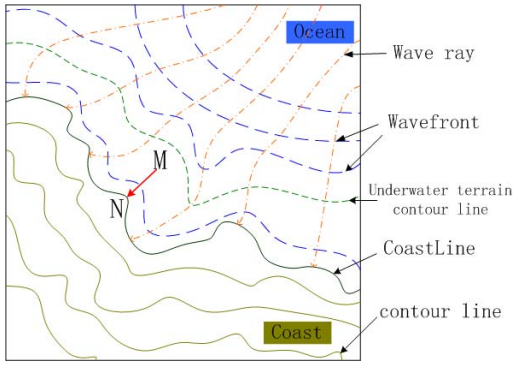


Figure 8. **Shallow ocean waves**

6. Results

This paper set an experimental environment as MS-Visual C++6.0 and DirectX9.0 SDK. PC is assembled with 1.6GHz Intel P4 CPU，512M system RAM and nVidia FX5600 with 128 MB of video memory. We use repeatable ocean surface blocks measuring 10 meters to 160 kilometers on a side, with the simulation resolution as high as 16385×16385 samples.

The algorithm of [11] has a good realistic effect, but only a small area of ocean and the frame rates are so slow. We compare our algorithm with his algorithm at same PC. The result shows that we have a more rapid rendering rate (see Figure 9). The most important is that our ocean surface is unbounded in deed. Some snapshots of ocean scene can be seen in Figure 10.

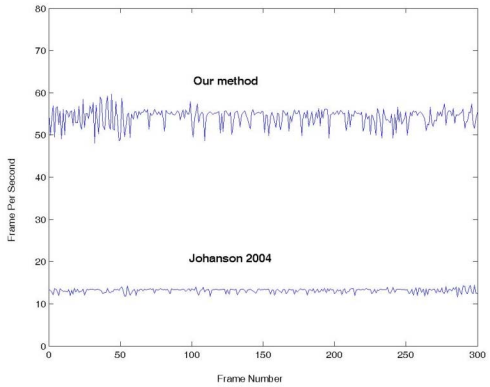


Figure 9. **Frame rates comparison between our method and [11]**

7. Summary and future work

This paper presents a framework for real-time simulation and rendering of unbounded ocean surface. We have been introduced the algorithm of Tiled Quad-tree and WFS into generating the multi-resolution mesh of ocean surface. Experimental results showed that our approach was effective and would satisfy the requirements of real-time navigation of unbounded realistic seascape.

The physical mechanism of ocean is very complex, so we have a long way to go. In the future, curling and breaking waves, various effects of ocean waves near the shore, such as foam, spray and ripple etc, should have been paid more attention in our work.

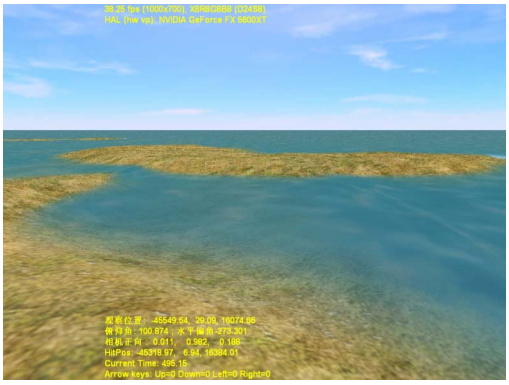
8. Acknowledgements

This research was supported partially by National “973” Key Project of Fundamental Research of China (No.2002CB312105) and National “863” High Technology Plan Foundation of China (No. 2004AA115130).

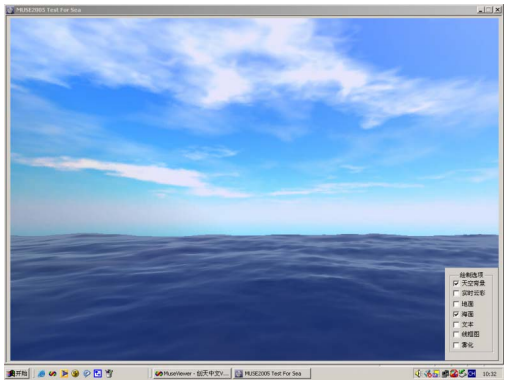
9. References

[1] Perchy D R., Modeling wave and surf. [J], Computer Graphics, 1986, 20(4): 75-83.
[2] Jim X. Chen. Physically-based modelling and real-time simulation of fluids. PhD in the Department of Computer Science of University of Central Florida,1995.
[3] Simon Premoze, Michael Ashikhmin, Rendering natural waters, Proceedings of Pacific Graphics 2000:23-30.
[4] Perlin, K. An image synthesizer. In Computer Graphics (SIGGRAPH ’85 Proceedings), B. A. Barsky, Ed., 1985, vol. 19(3), 287–296.
[5] Thon, S., Dischler,J.-M., and Ghazanfarpour, D. Ocean waves synthesis using a spectrum-based turbulence function. In Computer Graphics International Proceeding. 2000.
[6] Joe Stam, A simple fluid solver based on the FFT, Journal of Graphics Tools. 2001,6(2): 43-52.
[7] Foster N, Fedkiw R. Practical animation of liquids, [J]. SIGGRAPH’ 2001: 15-22.
8] Gary A. Mastin, Peter A. Watterberg, and John F. Mareda. Fourier synthesis of ocean scenes. IEEE Computer Graphics and Applications, March 1987, pages 16–23.
[9] Pauline Y. Ts’o and Brian A. Barsky. Modeling and rendering waves: Wave-tracing using beta-splines and reflective texture mapping. ACM Transactions on Graphics, 1987,6(3): 191–214.
[10] Kass.Ram, M G Rapid. Stable fluid dynamics for computer graphics, Computer Graphics, 1990,24(4): 49-57.
[11] Claes Johanson, Real-time water rendering, thesis of master, Department of Computer Science, Lund University, 2004.
[12] Damien Hinsinger, Fabrice Neyret and Marie-Paule Cani, Interactive Animation of Ocean Waves, ACM Symposium on Computer Animation, 2002,161–166.
[13] Vladimir Belyaev, Real-time simulation of water surface, Applied Math. Department, St. Petersburg State Polytechnical University, St. Petersburg, Russia.
[14] Layton, A.T. Numerically Efficient and Stable Algorithm for Animating Water Waves. The Visual Computer, 2002,Vol. 18, No.1, pp. 41-53,.
[15] Manuel N.Gamito and F.Kenton Musgrave, An Accurate Model of Wave Refraction Over Shallow Water, Computers & Graphics, 3 June 2002.
[16] Stefan Jeschke, Hermann Birkholz and Heidrun Schmann, A Procedural Model for Interactive Animation of Breaking Ocean Waves, WSCG’2003, February 3-7, 2003, Plzen, Czech Republic.
[17] Jensen, L. S., and Golias, R. Deep-Water Animation and Rendering. In Gamasutra September 2001.
[18] Changbo Wang, Zhangye Wang, Jianqiu Jin,and Qunsheng Peng. Real-time Simulation of Ocean Wave Based on Cellular Automata. CAD/Graphics’2003 October 29-31, Macao, China.
[19] Tessendorf, J., Simulating ocean waters. In SIGGRAPH course notes (course 47), ACM SIGGRAPH, 2001. http://home1.get.net/tssndrf/.

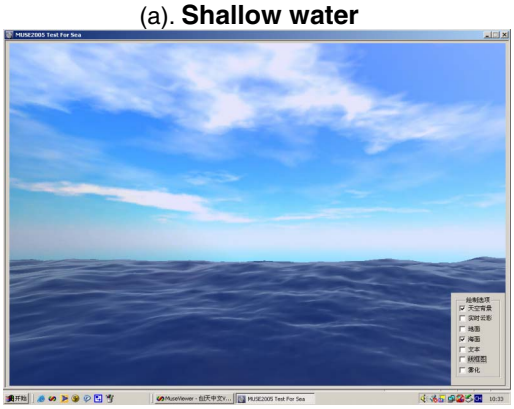
[20] Cignoni, P., Ganovelli, F., Gobbetti, E., Martom, F., Ponchio, F., and Scopigno, R. BDAM – Batched dynamic adaptive meshes for high performance terrain visualization. Computer Graphics Forum 22(3), 2003.
[21] Levenberg, J. Fast view-dependent level-of-detail rendering using cached geometry. IEEE Visualization 2002, 2002, 259-266.
[22] Wagner, D. Terrain geomorphing in the vertex shader. In ShaderX2: Shader Programming Tips & Tricks with DirectX 9. Wordware Publishing. 2004.
[23] Zhao Youbing, Zhou Ji, Shi Jiaoying and Pan Zhigeng. A fast algorithm for large-scale terrain walkthrough. In Proceedings of CAD/Graphics'2001. Kunming: 2001.
[24] Thatcher Ulrich. Rendering Massive Terrains Using Chunked Level of Dedail Control, SIGGRAPH 2002 course.
[25] Jason L. Mitchell. Real-Time Synthesis and Rendering of Ocean Water, ATI Research Technical Report, April 2005.



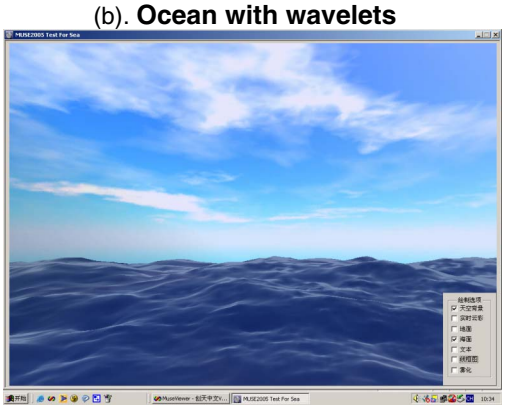
(a). **Shallow water**



(b). **Ocean with wavelets**



(c). **Ocean with middle waves**



(d). **Ocean with billowy waves**

Figure 10. **Snapshots of ocean scene. In figure (a) the display fps is less 40 because the rendering of terrain wastes the time.**