An Efficient Method for Real-Time Ocean Simulation

Haogang Chen, Qicheng Li, Guoping Wang,
Feng Zhou, Xiaohui Tang, and Kun Yang

Dep. of Computer Science & Technology, Peking University, Beijing, China
chg@geoagent.pku.edu.cn
{lqc,wgp,zf,txh,yk}@graphics.pku.edu.cn

Abstract. Ocean stimulation is important for computer animation, games, and other virtual reality applications. In this paper, a real-time method for simulating integrated ocean environment, including sky sphere, atmospheric system and ocean wave model, is proposed. By this method, the effect of cloud floating in sky sphere is obtained through texture perturbation, while the atmospheric system realizes the air scattering and absorbing effect. The ocean wave model constructs ocean surface mesh by Sine wave and realizes the bumping effect of ocean surface through normal disturbance of bump map method. For the lightening computation of ocean wave, the incident ray can be obtained by sampling the sky sphere and the reflecting light ray can then be calculated through the principle of mirror reflection. The proposed algorithm can be easily accelerated by GPU hardware. Experiments show that the method is easy to implement, and is effective to render ocean at high frame rate

1 Introduction

The modeling and rendering of water has been a traditional problem in computer graphics during the last two decades. It has always been a topic of many applications, such as games and animation. In particular, ocean water is difficult to be simulated due to the shape and combination of multiple waves, in addition to the reflection of the clouds and the sun in the sky.

Recently some efforts have been paid on the concept and the different levels of realism (physical realism, photo realism and functional realism) within the framework of computer graphics [1] and fluid visualization [2]. In this paper, our focus is the techniques for simulating ocean, but not those intended for fluid visualization. Therefore, a functional realism for ocean simulation, being more effective and easy than physical realism, is utilized in our work. That is to say, we are simply concerned with generating a visually convincing appearance of motion whether or not this motion actually follows the laws of physics. Of course, this computer graphics approach sometimes yielded unrealistic effects, making it unsuitable for real experiments. However, it is still quite useful in computer games and entertainment since very beautiful images and animations can be produced. As Fournier and Reeves said [3]:

We do not expect any "physical" answer from the model. In fact, we do not ask any question except "does it look like the real thing".

The paper is organized as following. In Section 2 some related work is reviewed. Then, in Section 3, we present the theory and implementation of our ocean environment model. In Section 4 and 5, experimental results are introduced, followed by conclusions.

2 Related Work

Here we do not review those work dedicated to running water or to rivers such as [4, 5, 6, 7, 8], which focus on other kinds of water surfaces and are not adapted to the simulation of ocean waves.

In the middle of 1980s Fournier and Reeves [3] proposed an ocean wave model based on the Gerstner–Rankine model. Roughly speaking, this model establishes that particles of water describe circular or elliptical stationary orbits. Peachey [9] proposed a similar idea, with fewer refinements (e.g., no trochoids). More recently, several models [10, 11] have proposed more precise ways to solve the propagation (wave tracing). Note that noise is generally used in all the models above in order to avoid the visual regularity due to the fact that only one or two wave trains are simulated. The spectral approaches are first introduced in CG by Mastin et al. [12]. The basic idea is to produce a height field having the same spectrum as the ocean surface. This can be done by filtering a white noise with Pierson-Moskowitz's or Hasselmann's filter, and then calculating its Fast Fourier Transform (FFT). The main benefit of this approach is that many different waves are simultaneously simulated, with visually pleasing results. However, animating the resulting ocean surface remains challenging. Tessendorf [13] shows that dispersive propagation can be managed in the frequency domain and that the resulting field can be modified to yield trochoid waves. A positive property of FFTs is the cyclicity: the solution can be used as a tile, which allows to enlarge the usable surface region as long as the repetitiveness is not obvious. The corresponding negative aspect of FFTs is homogeneity: no local property can exist, so no refraction can be handled. It should be noted that this model is the one implemented by Areté Image Software and used for the special effects of many movies such as ***Waterworld*** or ***Titanic***.

In recent years, with the development of hardware, today's commodity graphics hardware begins to provide programmability both at the fragment level and the vertex level. More and more efforts turn to graphics processing unit (GPU) to solve general-purpose problems [14]. And a lot of GPU-based ocean simulation method has been brought out. ATI Research presents a multi-band Fourier domain approach to synthesizing and rendering deep-water ocean waves entirely on the graphics processor[15]. Mark Finch [16] provides an explicit solution for simulating and rendering large bodies of water on the GPU. His method has proven suitable for real-time game scenarios, have been used extensively in Cyan Worlds' ***Uru: Ages Beyond Myst***. Yang adapts GPU acceleration to realize a multi-resolution mesh model of the ocean surface based on a straightforward terrain LOD scheme[17].

Based on such related work, our contributions are primarily due to: a) providing an integrated ocean environment model; b) improving the efficiency so that the ocean environment can be rendered at over 200 frame rate; c) shortening the implementation cycle because of the simplicity of our ocean model.

3 Ocean Environment Model

Our ocean environment model includes three parts: sky sphere, atmospheric scattering system and ocean wave model. As shown in Figure 1, sky sphere realize a wispy clouds effect, and the atmospheric scattering model computes the incident ray from the sky sphere to the ocean wave. Then we model the ocean wave surface, and compute the reflex ray based on the law of refection.

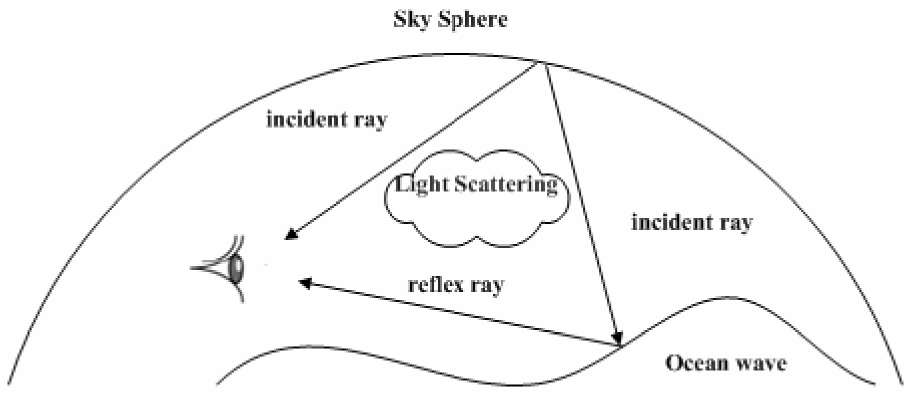


Fig. 1. The ocean environment model

3.1 Sky Sphere

The recent generation of graphics hardware has brought about a new level of per-pixel programmability for real time graphics, especially with regards to dependant texture reads. With the ability to perform per-pixel math both before and after texture fetches, a variety of new texture based effects are now possible. Texture perturbation [18] is that use the results of a texture fetch to change the texture coordinates used to fetch another texel. Here we adopt texture perturbation effect to simulate the wispy cloud. The idea is to scroll two tileable textures past each other in opposite directions. The first texture is the perturbation texture, which has u and v perturbations stored in the red and green channels of the texture. The second texture is the cloud texture, which has its texture coordinates perturbed by the perturbation map. In order to make the clouds look much more realistic however, its best to use multiple layers of clouds. The cloud layers are scrolled past each other at different speeds to give the effect of depth through parallax.

3.2 Atmospheric Scattering

The atmospheric scattering effect plays a significant role at the render of outdoor scenes. It directly decides the color and lightness of sky. However, it also woks on the color of the objects in the scene, and make users feel the distance and hierarchy of scene objects, and this is the most effect of atmospheric scattering.

The traditional outdoor render always uses fog model in hardware to simulate the effect. An equation can express the fog model:

$$L = L_0(1 - f) + C_{fog} f \tag{1}$$

Where L_0 is the intrinsic color, C_{fog} is the color of fog, and f is the fog coefficient. This model has three deficiencies: first, the fog coefficient f is monochrome, but actually scattering influence the rays at various wavelength differently; secondly, f, being a function of distance, can not express the physical principle of scattering, so the precision of the model can not be guaranteed; finally, the color and lightness of C_{fog} can not change based on viewing direction. Therefore, a model provided by Hoffman which is more physically accurate [19] is adopted in this work.

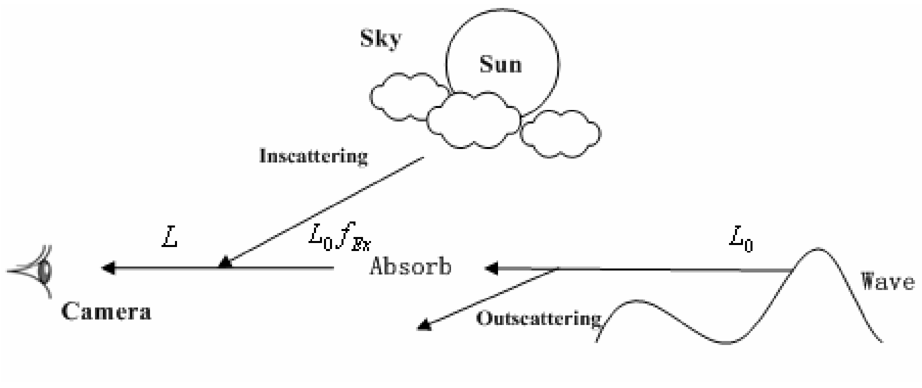


Fig. 2. Atmospheric scattering model

Based on the physical model, we can get this equation as follows:

$$L = L_0 f_{Ex} + L_{In} \tag{2}$$

where f_{Ex} is the term of attenuation, L_{In} is the incidence ray from sky. As shown in Figure 2, we can see that L_0 which is the reflex ray with intrinsic color is absorbed by atmospheric partly and deviate from primary direction as a result of outscattering, and these two parts compose the attenuation of ray. However, the ray from sky and other directions can radiate along the direction of the reflex ray, and this is the inscattering ray.

The equation of absorption and outscattering is that:

$$L(x) = L_0 e^{-\beta_{Ab} x} e^{-\beta_{Sc} x} = L_0 e^{-(\beta_{Ab} + \beta_{Sc}) x} \tag{3}$$

where β_{Ab} is the factor of absorption, β_{Sc} is the factor of scattering. β_{Ab} and β_{Sc} are based on the wavelength of ray, and in RGB system they can express as a triple vector. In practice, the β_{Ab} is set to the value suggested in [20], and β_{Sc} is the sum of Rayleigh coefficient and Mie coefficient.

However there is also phenomena which can add light to a ray. One case is in-scattering, where light which was originally headed in a different direction is scattered into the path of a light ray and adds to its radiance. For computing the inscattering ray, we define the scattering phase function Φ(θ) which gives the probability of scattered light going in the direction (θ), θ is the angle of incidence ray and scattering ray. Since this is a probability function, when integrated over the entire sphere of directions the result is 1. So we get this equation:

$$\frac{\Delta L}{\Delta x} = -\beta_{Ex} \cdot L + E_{Sun} \cdot \phi(\theta) \cdot \beta_{Sc} \tag{4}$$

E_{Sun} is a scalar factor and express the energy of sun.

From (4), we can get:

$$L(x) = E_{Sun}(L_0 e^{-\beta_{Ex} x} + \frac{\beta_{Sc}\phi(\theta)}{\beta_{Ex}}(1 - e^{-\beta_{Ex} x})) \tag{5}$$

Since β_{Sc} is the sum of Rayleigh coefficient and Mie coefficient, the function $\phi(\theta)$ depends on the particle doing the scattering. For particles much smaller than the wavelength of light (r < 0.05 λ), $\phi_R(\theta)$ is based on Rayleigh coefficient. For larger particles, $\phi_{HG}(\theta)$ is based on the Henyey / Greenstein function which is an approximation to the Mie coefficient.

$$\phi_R(\theta) = \frac{3}{\pi}(1 + \cos^2\theta) \tag{6}$$

$$\phi_{HG}(\theta) = \frac{(1-g)^2}{4\pi(1 + g^2 - 2g\cos(\theta)\ )^{3/2}} \tag{7}$$

g is the eccentricity of Mie coefficient, we use an approximation to express it, which scales with $1/\lambda^2$. Hence, the equation of $\beta_{Sc}\phi(\theta)$ is that:

$$\beta_{Sc}\phi(\theta) = \beta_R \frac{3}{\pi}(1 + \cos^2\theta) + \beta_M \frac{(1-g)^2}{4\pi(1 + g^2 - 2g\cos(\theta)\ )^{3/2}} \tag{8}$$

Substitute the $\beta_{Sc}\phi(\theta)$ of expression (5) with expression (8), we can wok out the inscattering ray.

3.3 Ocean Wave

Our ocean system combines geometric undulations of a base mesh with generation of a dynamic normal map. First, we generate the combination of sine waves to form the ocean wave mesh, and produce normal disturbance on the ocean wave mesh based on bump map [22]. Then the sky sphere is sampled to get the incidence ray. Finally the reflect ray is obtained based on Fresnel reflection formula. Obviously this ocean wave model is very effective, it is easy to form a ocean wave surface, and for using normal bump maps to generate composite normals on each mesh face, the ocean grid can be very coarse, but a high sense of reality can be generated. For example, Tessendorf [13] form an ocean surface based on FFT, and need sample a 2048*2048 grid for building an ocean environment; while, the system that we realize only need sample a 50*50 grid. To obtain more reality of ocean scene, we realize the atmospheric scattering model, and calculate the scattering effect for the reflex ray of ocean wave.

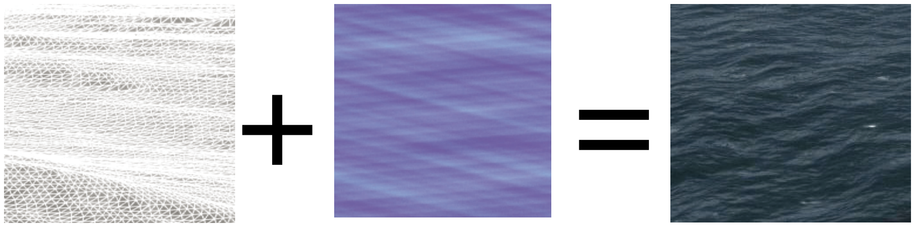


Fig. 3. Ocean wave model

The algorithm is implemented on GPU using vertex shader and pixel shader. The vertex shader is responsible for generating the combination of sine waves that perturb the position. First, a height map is used to allow artist control of how high the waves are in different parts of the ocean. This can be useful for shorelines where the ocean waves will be smaller than those further out to sea; Second, the height of every position is computed based on the input sine waves, and due to the SIMD nature of vertex shaders, 4 sine waves are calculated in parallel; for perturbing the bump map's tangent space normal, vertex shader also need compute the transform matrix from the tangent space into world space. The pixel shader is responsible for producing the bump-mapped reflective ocean surface.First, it transforms the tangent space composite normal into world space and calculates a per-pixel reflection vector Second, the reflection vector is used to sample the sky sphere;At last, the reflex ray is computed based on Fresnel reflection formula.

$$\mathrm{C}_{\mathrm{water}} = F * C_{reflect} + (1 - F) * C_{refract} \tag{8}$$

$$F = \frac{(g-k)^2}{2(g+k)^2}\left(1 + \frac{(k(g+k)-1)^2}{(k(g-k)+1)^2}\right)(k = \cos\alpha, g = n_a / n_b + k^2 - 1)$$

Where F is the Fresnel factor, α is the angle of incidence, n_a and n_b are the refractive index of air and water.

The color of the reflex ray is not the final color, because we calculate the atmospheric scattering model, and the reflex ray is the initial ray L_0 of the atmospheric scattering model, so we must compute the absorption and scattering effect for the reflex ray.

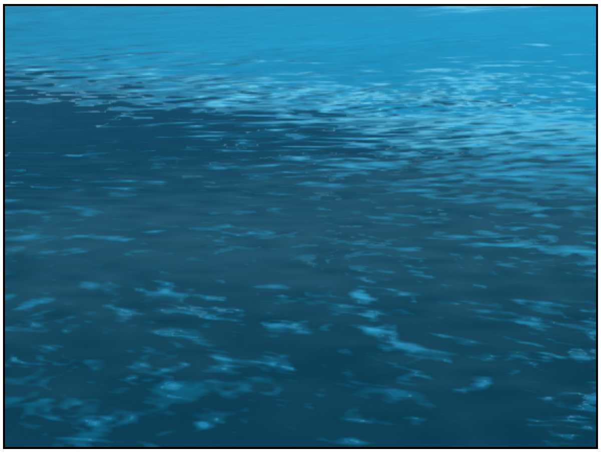
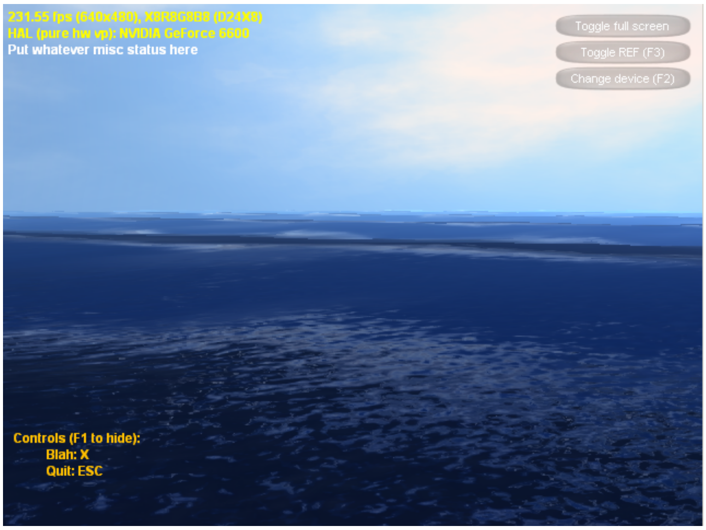


Fig. 4. The ocean wave

4 Results

Since mostly computation is carried on the GPU, the performance of proposed method is relatively sensitive with the GPU power. Our experimental PC is assembled with 1.6GHz Intel P4 CPU, 512M system RAM and nVidia GeForce 6600 with 128 M of video memory and the test program window is at the resolution 640*480. With this environment the frame rate of render ocean scene is over 200. Another experimental PC is assembled with AMD Athlon(TM) XP 1800+ CPU, 512M system RAM and ATI RADEON 9550 with 128M of video memory. With this environment the frame rate of rendering ocean is 100 almost.



```
231.55 fps (640x480), X8R8G8B8 (D24X8)
HAL (pure hw vp): NVIDIA GeForce 6600
Put whatever misc status here
```

```
Controls (F1 to hide):
    Blah: X
    Quit: ESC
```

Fig. 5. The integrated ocean environment

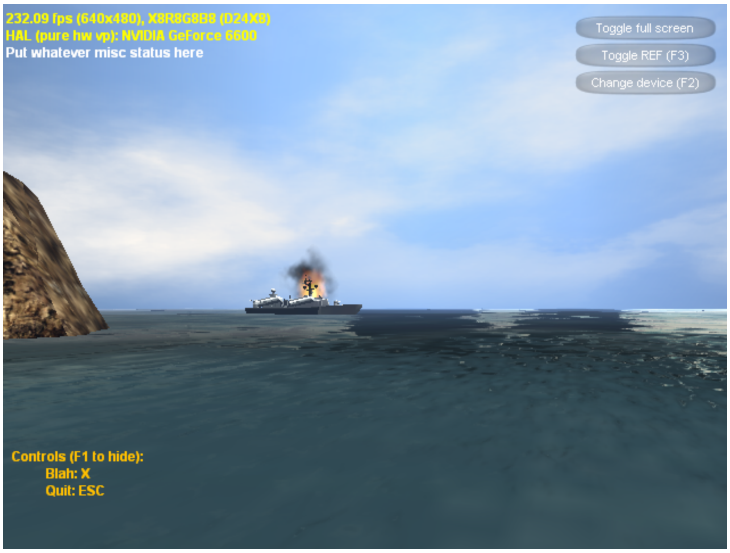

232.09 fps (640x480), X8R8G8B8 (D24X8)
HAL (pure hw vp): NVIDIA GeForce 6600
Put whatever misc status here
Toggle full screen
Toggle REF (F3)
Change device (F2)
Controls (F1 to hide):
Blah: X
Quit: ESC

Fig. 6. The image of ocean war

Our method has been implemented in an ocean war simulation platform, we also use the texture perturbation effects to simulate the fire and smoke. Snapshots of ocean scene are showed in Figure6.

5 Conclusion and Future Work

A novel ocean environment model, including sky sphere, atmospheric system and ocean wave model, is proposed in this paper. Texture perturbation is employed on the sky sphere mesh to realize wispy clouds. We specifically give the explanation that how atmosphere system affects light and realize the absorption and scattering effect. Also we present an ocean wave model, which combines sine waves to form the ocean wave mesh, and produce normal disturbance on the ocean wave mesh based on bump map. Our algorithm is accelerated by GPU hardware, which makes the method easy to implement and efficient in rendering with high sense of reality.

Although being with several advantages, this method has some deficiencies. Since we use sine waves to simulate ocean wave, the ocean wave is seasonal. Secondly, we do not consider the wakes of objects, and the effect is important for simulate the sail of warship. These issues will be in-depth addressed in our future work.

Acknowledgement

This research was supported by the National Grand Fundamental Research 973 Program of China (Grant No. 2004CB719403), and the National High Technology Development 863 Program of China (Grant No. 2004AA115120).

References

1. J.A. Ferwerda, Three varieties of realism in computer graphics, in: Cornell Workshop on Rendering, Perception and Measurement, 1999.
2. A. Iglesias Computer graphics for water modeling and rendering: a survey , Future Generation Computer Systems, 20(8), Nov. 2004.
3. A. Fournier, W.T. Reeves, A simple model of ocean waves, in: Proceedings of SIGGRAPH'86, Comput. Graph. 20 (4) (1986) 75–84.
4. Kass, M., and Miller, G. Rapid, stable fluid dynamics for computer graphics. In Computer Graphics (SIGGRAPH '90 Proceedings), F. Baskett, Ed., vol. 24, 49–57.
5. Foster and Metaxas, D. Realistic animation of liquids. In Graphics Interface '96, W. A. Davis and R. Bartels, Eds., 204–212.
6. Foster, N., and Fedkiw, R. Practical animation of liquids. Proceedings of SIGGRAPH 2001 (August), 23–30.
7. Neyret, F., and Praizelin, N. Phenomenological simulation of brooks. In Eurographics Workshop on Computer Animation and Simulation, Springer, Eurographics, 53–64. 2001.
8. Thon, S., and Ghazanfarpour, D. A semi-physical model of running waters. In Eurographics UK. 2001.
9. D.R. Peachey, Modeling waves and surf, in: Proceedings of SIGGRAPH'86, Comput. Graph. 20 (4) (1986) 65–74.
10. TS'O, P., and Barsky, B. Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping. ACM Transactions on Graphics 6, 3 (July), 191–214. 1987.
11. GONZATO, J.-C., and SAËC, B. L. On modelling and rendering ocean scenes. The Journal of Visualization and Computer Animation 11, 1, 27–37. 2000.
12. MASTIN, G. A., WATTERBERG, P. A., AND MAREDA, J. F.Fourier synthesis of ocean scenes. IEEE Computer Graphics and Applications 7, 3 (Mar.), 16–23. 1987.
13. J. Tessendorf, Simulating ocean water, in: SIGGRAPH'99 Course Notes, 1999.
14. GPGPU http://www.gpgpu.org/
15. Jason L. Mitchell, "Real-Time Synthesis and Rendering of Ocean Water," ATI Technical Report, April 2005.
16. Finch M. "Effective Water Simulation from Physical Models," GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics. Edited by Randima Fernando, Page 5-29. 2004.
17. Xudong Yang, Xuexian Pi, Liang Zeng, Sikun Li, "GPU-Based Real-time Simulation and Rendering of Unbounded Ocean Surface" Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG'05), Hong Kong, Dec 07-10 2005.
18. John Isidoro and Guennadi Riguer, "Texture Perturbation Effects, " ShaderX, WordWare Inc., 2002,ISBN 1-55622-041-3.
19. Naty Hoffman, Nathaniel and Arcot J. Preetham, Rendering Outdoor Light Scatter in Real Time, Proceedings of Game Developer Conference 2002.
20. A.J.Preetham, P.Shirley, B.E.Smit. A Practical Analytic Model for Daylight. Computer Graphics(Proceedings of SIGGRAPH 1999):91-100, August 1999.
21. Jan Kautz, Wolfgang Heidrich, Hans-Peter seidel. Real-time bump map synthesis. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware Aug.2001.