Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9034649, 9 pages
http://dx.doi.org/10.1155/2016/9034649

Research Article

Automatic Optimization for Large-Scale Real-Time Coastal Water Simulation

Shunli Wang[1,2] and Fengju Kang[1,2]

[1]*School of Marine Technology, Northwestern Polytechnical University, Xi'an, China*
[2]*National Key Laboratory of Underwater Information Process and Control, Xi'an, China*

Correspondence should be addressed to Shunli Wang; wangsli2011@126.com

Received 6 April 2016; Accepted 10 October 2016

Academic Editor: Ruben Specogna

Copyright © 2016 S. Wang and F. Kang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We introduce an automatic optimization approach for the simulation of large-scale coastal water. To solve the singular problem of water waves obtained with the traditional model, a hybrid deep-shallow-water model is estimated by using an automatic coupling algorithm. It can handle arbitrary water depth and different underwater terrain. As a certain feature of coastal terrain, coastline is detected with the collision detection technology. Then, unnecessary water grid cells are simplified by the automatic simplification algorithm according to the depth. Finally, the model is calculated on Central Processing Unit (CPU) and the simulation is implemented on Graphics Processing Unit (GPU). We show the effectiveness of our method with various results which achieve real-time rendering on consumer-level computer.

1. Introduction

Water is a common and important element in nature. It is also frequently encountered in games and other 3D virtual environments. A lot of attention has been paid to rendering water realistically, but the effectiveness of large-scale coastal water is typically limited in games. Large-scale water is often simply modeled to decrease computer cost, so there are not enough details, such as surf, foam, spray, and so forth. More details are needed to ensure the visual effect in today's computer games and 3D virtual environments.

Methods of coastal water simulation can be classified into physically based methods and geometry based methods. Physically based methods mainly solve the shallow-water equations (SWE) with a grid-based or particle-based solver, which requires more than thousands of grid cells or particles [1]. A typical grid-based simulation is developed by Playne et al. [2]. They solve the SWE on multiple graphics processing units (GPUs) and give a standard parallel scheme for realistic fluid simulations. A hybrid grid which consists of regular cells near the water surface and tall cells on the bottom is given by Chentanez and Müller [3]. This method can handle most water phenomena while reducing computation time. It is implemented on a GPU and can get a good real-time performance. Ojeda and Susin [4] simulated shallow-water with the algorithm based on the Lattice Boltzmann Method (LBM). They enhanced the shallow-water scene [5] by adding lower details, surface foam, caustics, and so forth. A fast shallow-water is developed by Abbasov [6]. He constructed a discrete model after numerical simulating a three-dimensional (3D) shallow-water model. This method can greatly simplify the computation by adjusting the cell fill factor upon a sloping shore. The particle-based methods are more suitable for simulating small-scale water with higher details, such as breaking waves [7], water flows into empty regions [8], and interaction of water and rigid objects [9]. Geometry methods usually use a series of shapes to simulate shallow-water. They have been used for a long time. Gerstner model is applied by Fournier and Reeves [10] to the computer graphics field and improved with the elliptic function. However, the water shape he obtained cannot be curved in his simulation. A procedural model for breaking wave in coastal water is provided by De Lima et al. [11]. They generate a wave map and add it to water surface. This method can efficiently simulate the coastal water. Luo et al. [12] put forward a kind of 3D Gerstner wave model with the 3D Bezier curved. These methods

can generate a realistic coastal water scene, but they still cannot meet the real-time requirement when large bodies of coastal water need to be rendered. A hybrid shallow-water that combines grid-based and particles based simulation is provided by Chentanez and Müller [13]. Longo et al. [14] gave a review for turbulence in the swash zone and discussed the measurement techniques and generation of breaking waves. There also have shallow-water simulations using a depth-integrated [15], spectrum [16, 17], 2D Boussinesq equations [18], video-based method [19], and so forth. The geometry based methods are suitable for large-scale real-time water because of simple computation. However, these methods cannot simulate the vertical movement, such as curved waves and breaking waves. Although the above methods can get a nice shallow-water scene, they have some limits to meet the requirements of large-scale real-time shallow-water simulations.

In order to realize large-scale real-time shallow-water simulations, we analyze the coastal water modeling, height field map generation, ocean wave mesh generation, and rendering. We find some solutions to enhance the real-time performance of the simulations. Our contributions of this paper are as follows:

(i) A construction of the hybrid model that allows the simulation of arbitrary domain scales by coupling of deep water and shallow-water waves.

(ii) An adaptive optimization of water surface grid that can efficiently reduce unnecessary grid cells under the coastal terrain.

(iii) Implementations of our method and water surface effects, such as surface foam and reflection, with the shader language on GPU.

This paper is organized as follows. We present our hybrid model in the next section. The optimization of water surface grid is shown in Section 3. In Section 4, we present the implementation of our method. Then, we show our rendering results and discussion in Section 5. Finally, we conclude and explore avenues of future work in Section 6.

2. Hybrid Coastal Water

When the terrain depth (Depth, D) is greater than half of the wavelength (L), the water is called deep water; otherwise, the water is called shallow water when D is less than $L/20$, and the intermediate zone shown in Figure 1 is the transition zone.

To achieve a realistic coastal water scene, this paper selects the deep water model based on FFT [20] and the shallow-water model based on geometry [21], and the automatic fusion algorithm is used to couple the two water waves in the transitional zone.

2.1. Deep Water. The FFT method based on Phillips spectrum first gets a series of double frequency sine waves whose amplitude and phase are relevant and then gets the height of every grid point on the ocean face by calculating with FFT.

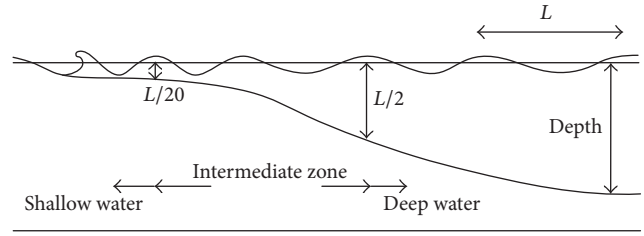


Figure 1: The intermediate zone from deep water to shallow-water.

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

$$h(\mathbf{x}, t) = \sum_{\mathbf{k}} h(\mathbf{k}, t)\, e^{i\mathbf{k}\cdot\mathbf{x}}, \tag{1}$$

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

$$P_h(\mathbf{k}) = A\frac{1}{k^4} e^{-1/(kL)^2} \left|\widehat{\mathbf{k}} \cdot \widehat{\omega}\right|^2. \tag{2}$$

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

$$h_0(\mathbf{k}) = \frac{1}{\sqrt{2}} (\xi_r + i\xi_i) \sqrt{P_h(\mathbf{k})}, \tag{3}$$

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

$$h(\mathbf{k}, t) = h_0(\mathbf{k})\, e^{i\omega(k)t} + h_0^*(-\mathbf{k})\, e^{-i\omega(k)t}, \tag{4}$$

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

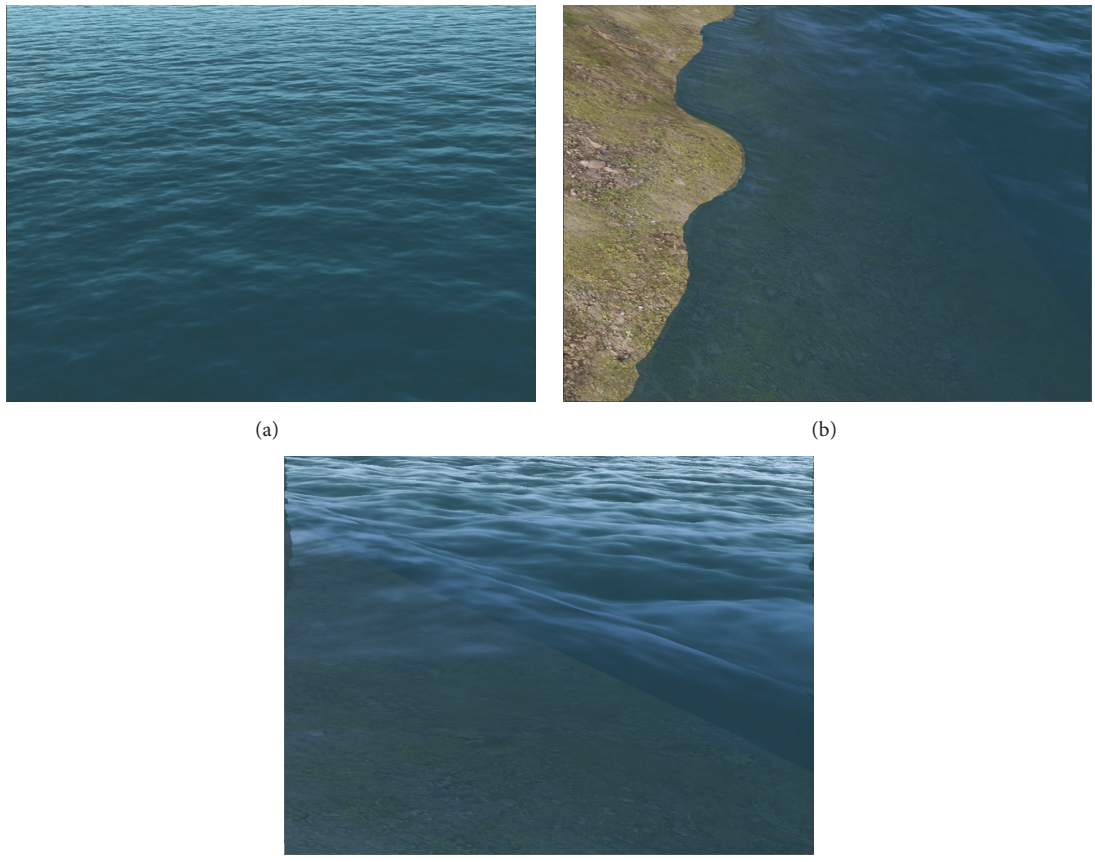
Actually, in real motion of an ocean wave, the continuous changes of wind direction lead to extrusion between waves which make it become sharp at the top of waves and flat at the bottom following the rolling wave effect. The height map generated by our method is the result of stacking of multiple sine waves. 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 the grid in the horizontal direction. Here we use the amplitude of the ocean wave to get the offset of discrete point $\mathbf{x}$ in a horizontal direction:

$$D(\mathbf{x}, t) = -\sum_{\mathbf{k}} \left(i\frac{\mathbf{k}}{k}\right) h(\mathbf{k}, t)\, e^{i\mathbf{k}\cdot\mathbf{x}}. \tag{5}$$

But this method needs one more FFT, and we simplify it for accelerated computing:

$$D(x, y) = \frac{S}{N}\left(M_{x+1,y} - M_{x,y}, M_{x,y+1} - M_{x,y}\right), \tag{6}$$

where S is the size of the FFT grid, N is the number of FFT grid, and M is the wave-height map. The deep water simulation based on FFT is shown in Figure 2(a).



(a) (b)

(c)

Figure 2: The visual results of deep water (a), shallow-water (b), and the hybrid water (c) after coupling.

2.2. Shallow-Water. Compared to the deep water, shallow-water is more complicated due to the effects of seafloor terrain. Its crest line and amplitude will change as the depth decreases. This greatly increases the difficulty of water modeling and rendering. On account of the real-time requirement, this paper uses an improved Fournier model based on geometry instead of the physically based model.

Fournier proposed that the motion trails of water points gradually change from circular to elliptical with the decreasing of seabed depth based on the classical Gerstner model. He established the Fournier wave model and applied it to computer graphics. And then he realized the 3D visualization of coastal water. For the sake of rendering of coastal water, including curved waves, Wang et al. [21] made an improvement of Fournier's model and could quickly simulate coastal water and curved waves at different wind speeds. The formula is as follows:

$$\begin{aligned} x &= x_0 + R\cos(\alpha) S_x \sin(\phi)\cos(\theta) \\ &\quad + R\sin(\alpha) S_z \cos(\phi)\cos(\theta) + \Delta x \cos(\theta), \\ y &= y_0 + R\cos(\alpha) S_x \sin(\phi)\sin(\theta) \\ &\quad + R\sin(\alpha) S_z \cos(\phi)\sin(\theta) + \Delta x \sin(\theta), \\ z &= z_0 - R\cos(\alpha) S_z \cos(\phi) + R\sin(\alpha) S_x \sin(\phi) \\ &\quad + \Delta z. \end{aligned} \tag{7}$$

Among them, the plane (x, z) is a vertical surface wave, the point (x_0, z_0) is the static position of a wave point, R is the wave amplitude, ω is the angular frequency, $S_x = 1/1 - e^{-K_x h}$ and $S_z = S_x(1 - e^{-K_z h})$ are the major and minor axes of the ellipse, $\sin(\alpha) = \sin(\beta)e^{-K_0 h}$, β is the slope of seafloor terrain, K_∞ is the deep water wave number, $K(x) = K_\infty/\sqrt{\tanh(K_\infty h)}$ denotes the wave number at the point (x_0, z_0), $\phi = -\omega t + \sum_0^{x_0} K(x)\Delta x$, and θ is the wind direction.

In order to realize the curved waves, the wave parameters when moving need to be adjusted:

$$\begin{aligned} &\text{When } \phi_{\min} < \phi \le \phi_{\min} + \pi \\ &\quad \phi' = K_\phi \phi, \\ &\quad \Delta x = K_\phi x_0, \end{aligned}$$

$$\Delta z = 0,$$

$$S_x' = \frac{1}{2}\left[1 - \left(\frac{\phi - \phi_{\min} - \pi}{\pi}\right)^2\right] S_x. \tag{8}$$

When $\phi_{\min} + \pi < \phi \le \phi_{\min} + 2\pi$

$$\phi' = \phi,$$

$$\Delta x = \left(\frac{\phi - \phi_{\min} - 1.5\pi}{0.5\pi}\right)^2 (x_{\text{crest}} - x_{\text{curve}}), \tag{9}$$

$$\Delta z = \left(\frac{\phi - \phi_{\min} - 1.5\pi}{0.5\pi}\right)^2 (z_{\text{crest}} - z_{\text{curve}}),$$

$$S_x' = S_x.$$

And the limit on the long axis is

$$S_x \le f(x), \tag{10}$$

where $f(x) = 1/RK(x)\cos(\alpha)$ and η is the thin coefficient of the waveform. Its value is 1 by default. The higher the coefficient is, the thinner the waveform is. $\phi_{\min} = -\tan^{-1}((\sin(\alpha)S_x)/(\cos(\alpha)S_z))$ denotes the angular ϕ at the wave trough and K_ϕ is the curved coefficient with a minimum of 1.

The final coastal water is shown in Figure 2(b).

2.3. Coupling of Water Waves. In order to make the transition of deep water waves and the shallow ones smooth, we use the automatic coupling algorithm to couple the two waves when the depth is in the range of $0.05L$ to $0.5L$. The fusion algorithm is shown as follows:

$$h(x,y) = \begin{cases} h_{\text{FFT}}(x,y), & \text{if } d \ge 0.5L \\ \dfrac{d - 0.05L}{0.5L - 0.05L} h_{\text{FFT}}(x,y) + \dfrac{0.5L - d}{0.5L - 0.05L} h_{\text{geo}}(x,y), & \text{if } 0.05L < d < 0.5L \\ h_{\text{geo}}(x,y), & \text{if } d \le 0.05L, \end{cases} \tag{11}$$

where d is the terrain depth of the grid vertex at the current time. $h_{\text{FFT}}(x, y)$ and $h_{\text{geo}}(x, y)$ are the heights generated by the deep water and shallow-water model.

From the formula, it can be seen that the coupling algorithm will guarantee a smooth transition from deep water to shallow-water. The hybrid water after coupling is shown in Figure 2(c).

3. Water Surface Grid Automatic Simplification

When the water is rendered, grid model is usually used to simulate water surface. Yin et al. [22] divided the surface into a dynamic one and a static one. The dynamic surface is formed by $M * N$ grid points and revolves around the viewpoint, and the outer space of the dynamic surface is the static surface. Hinsinger et al. [23] presented an adaptive surface grid model. Cui et al. [24] improved the adaptive grid model and put forward a trapezoidal grid model. Johanson and Lejdfors [25] proposed a projection grid model and used it to simulate the water surface in world space.

The above studies of wave multiresolution grid did not take the influence of terrain into account. However, shallow-water will not appear under the land terrain, so the surface grid under the land should not be computed and rendered. In order to prevent the unnecessary expense, we proposed an improved multiresolution grid by detecting whether grid points are under the land. Then, rendering of coastal water will be accelerated after reducing computation consuming.

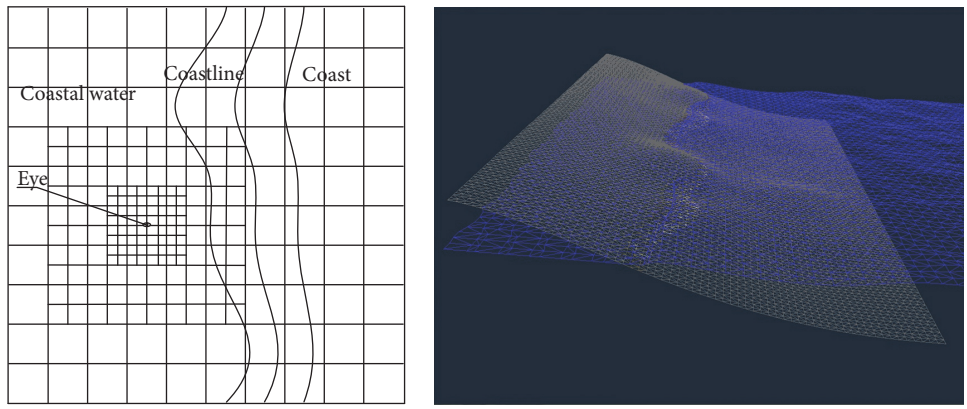
3.1. Multiresolution Grid. Multiresolution grid refers to grids with different levels of details (LOD) existing in different regions. It can greatly improve the efficiency of the grid generation without losing the details. Therefore, multiresolution grids are commonly used in real-time rendering of large-scales of water and terrain.

In the coastal scene rendering, a part of the water surface grid is always under the terrain. But the former researches usually use the same grid; they did not make changes to the grid. The general grid is shown in Figure 3.

The grid points covered by terrain will be calculated, generated, and automatically culled by the rendering system. There is no problem in vision, but the process of computing, generating, and culling can severely affect the real-time performance.

3.2. Grid Improvement. In order to avoid unnecessary computation, the grid points are detected in advance to ensure whether they are covered by terrain or not when the water surface grid is generating. If they are, the next computation step will be not processed; if not, then continue. The detection will last until the whole grid is generated. Because the coastal terrain is static, the detection does not need to be repeated. Then, the speed of generation and rendering of water can be improved.

In the case that coastline data are known, the cover situation can be judged according to the relationship between the grid points and the coastline. The detection process is as follows:



(a) (b)

Figure 3: The general multiresolution grid (a) and the rendering mesh (b).

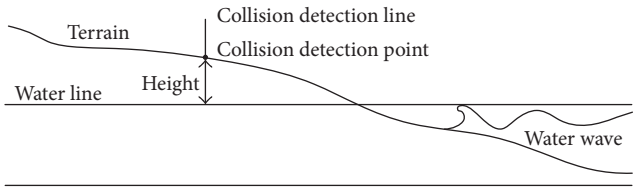


Figure 4: The collision detection process.

If $x_{\text{mesh}} < x_{\text{line}}$ & $y_{\text{mesh}} < y_{\text{line}}$, the grid point is not covered by terrain, or it is covered and will not need to be computed.

However, the coastline data cannot be generally obtained. Then, the collision detection technology will be used to detect whether the grid point is covered by terrain. The collision detection process is shown in Figure 4.

In the process, the collision detection system sends a vertical line from above the grid, and we can manage to keep an account of the height of the point relative to the horizontal line when the line collision with the terrain.

If $z_{\text{mesh}} < H$, the grid point is not covered by terrain, or it is covered and will not need to be computed. The final grid is given in Figure 5.

4. GPU Implementation Method

In order to rapidly generate large bodies of near ocean wave scene, graphics hardware shader language is used in rendering. The water height map data are obtained by the calculating of shallow-water model. Then, it is loaded in the vertex shader and is fused with Perlin noise. Final water height and offset in horizontal direction are computed. Each vertex normal vector and color data are calculated in the pixel shader, so the water scene is realized. The surface reflection and foam effect are also added to increase the scene fidelity.

4.1. Rendering Based on Height Field Map. The curved model can be computed to obtain the height and horizontal offset of each vertex. The result will be saved in a height map. Each pixel of the height map can store a vertex information. It is expressed by the formula

$$\text{color}_{\text{data}} \cdot rgb = (x, y, z). \tag{12}$$

The height field map data can be read and interpolated in the vertex shader, and Perlin noise is added; then, the final vertex information is expressed by

$$(x, y, z) = \left(\text{color}_{\text{data}} \cdot r, \text{color}_{\text{data}} \cdot g, \text{color}_{\text{data}} \cdot b + \text{Noise}_{\text{perlin}}\right). \tag{13}$$

In the pixel shader, the vertex normal vector and color are calculated, the vertex transparency is computed according to the depth:

$$\begin{aligned} \text{color}_{\text{final}} \cdot rgb &= \text{fresnel} * \text{color}_{\text{water}} \cdot rgb \\ &\quad + (1 - \text{fresnel}) * \text{color}_{\text{difuse}}, \\ \text{color}_{\text{final}} \cdot a &= 1 - 0.6e^{-h}. \end{aligned} \tag{14}$$

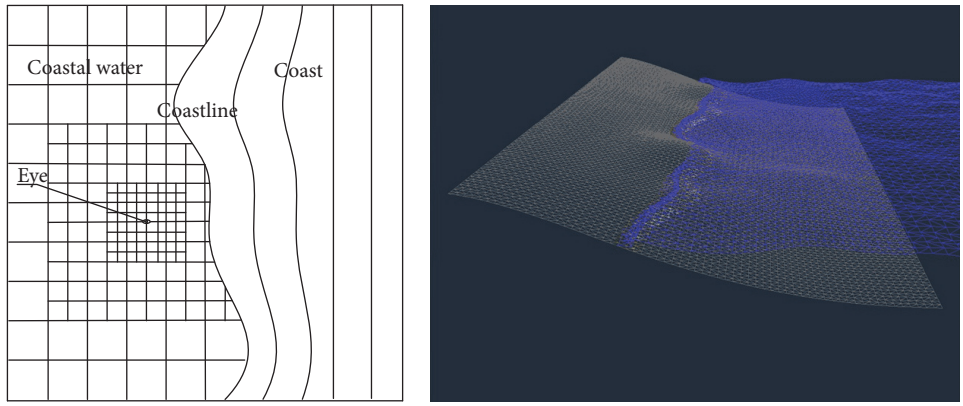
Here fresnel denotes the fresnel reflection coefficient.

The water surface will be getting more and more transparent with the decreasing of depth. The value of 0 means fully transparent. In this paper, the minimum transparency value is 0.6.

4.2. Realization of the Reflection and Foam Effect. The rendering scene above sea level is rendered to a texture through the RTT (render to texture) technology. The reflection texture is loaded in the pixel shader, and then the reflection effect of the wave surface is realized by using dynamic texture mapping technology. The specific calculation formula is as follows:

$$\begin{aligned} \text{color}_{\text{final}} \cdot rgb &= \text{fresnel} * \text{color}_{\text{water}} \cdot rgb \\ &\quad + (1 - \text{fresnel}) * \text{color}_{\text{reflect}}. \end{aligned} \tag{15}$$

Foam will be intermittently generated at the wave crest when the wave is moving. And foams appear in the trough



(a) (b)

Figure 5: The improved grid (a) and the rendering result (b).

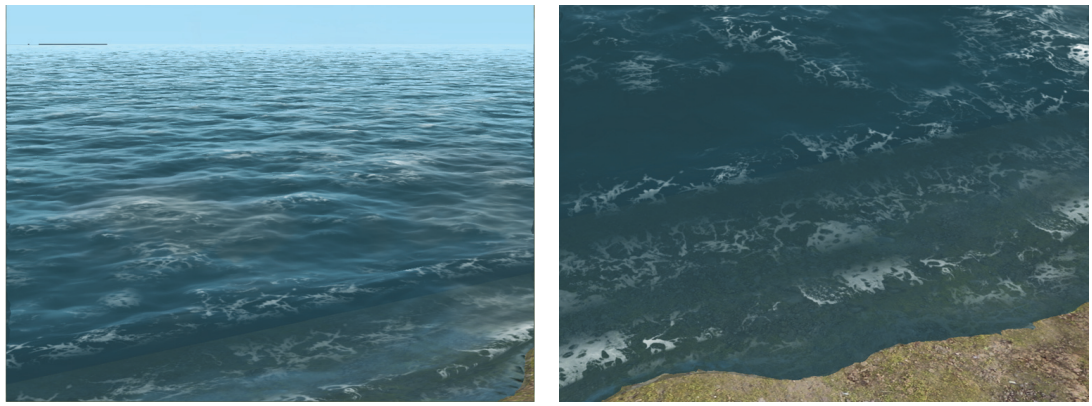


Figure 6: The reflection and foam effects.

and crest of the waves during wave run-up and run-down [14]. So we applied two different methods to achieve the effect of foam. We load a transparent foam texture and simulate the dynamic foam effect by controlling the offset speed of texture with texture mapping technology. The implementation detail and rendering result are shown in Box 1 and Figure 6:

$$\text{color}_{\text{final}} = \text{density} * \text{color}_{\text{final}} + (1 - \text{density}) * \text{color}_{\text{foam}}. \quad (16)$$

5. Results and Discussion

To evaluate our method, we have created several typical coastal water scenes on a 2.66 GHz Core i7 and NVIDIA GTX 560. The hybrid water model is computed on CPU, and the rendering is implemented on GPU. The results are shown in Figure 7. In Figures 7(a) and 7(b), the thin coefficient is 1.0, while its value is 1.2 in Figures 7(c) and 7(d). Compared with 1.0, the waveform is thinner with the coefficient of 1.2. But there is a problem that the waveforms will be crossed when the coefficient getting higher. It is the limit of our method, and we will research it in future work. The foam effects are different in the crest and trough of waves. In our method, foam at the crest moves forward and foam at the trough moves backward. So we can simulate the huge amount of bubbles fast. Table 1 lists the performance values of different examples. Since the model is geometric and the rendering is implemented with shader language, our method makes it possible to simulate a large-scale coastal water scene at a stable frame rate of 60 frames per second.

The comparison with the references' methods is shown in Table 2. Ojeda's method [4] is based on physics. It can handle arbitrary underlying terrain and interact with dynamic rigid bodies. They compute the water model both on CPU and on GPU, but only the result on GPU can meet the requirement of real-time performance. Another problem of our method is that it can only get the simulation of small-scale shallow-water. Luo's method [12] used a Bezier curve, so it is based on geometry. They can get a nice real-time middle-scale coastal water after GPU accelerating. However, they need to design a coastline before rendering. Once the rendering is begun, the coastline cannot be changed. As our method can get a high frame rate, we will not need a GPU based computation of

```
// Combine waterbody color and reflected color
vec4 water_color = mix(body_color, reflected_color, fresnel);
vec4 final_color = water_color;
float depth = texture2D(
                DeepMap, vec2(pos_local.x, (pos_local.y))/256.0 ).r * 30.0 - 15.0;
// Crest Foam
float alphaFactor = texture2D(
                FoamMap, pos_local.xy/64.0 + g_PerlinMovement / 2.0).r;
float foamFactor = alphaFactor * clamp(pos_local.z / 2.0 - 0.1, 0.0, 1.0);
final_color = vec4(final_color.xyz, (300.0 - pos_local.x) / 300.0 );
final_color = mix(final_color, vec4(1.0, 1.0, 1.0, 1.0), foamFactor);
// Trough Foam
alphaFactor = texture2D(
                FoamMap, vec2(pos_local.x/32.0, pos_local.y/32.0)
                + g_PerlinMovement / 2.0).r;
float foamFactor0 = alphaFactor * clamp( (depth + 3.0) / 5.0, 0.0, 1.0);
alphaFactor = texture2D(
                FoamMap, pos_local.xy/64.0 - g_PerlinMovement / 3.0).r;
float foamFactor1 = alphaFactor * clamp( -0.6 - pos_local.z, 0.0, 1.0)
                 * clamp( (depth + 20.0) / 15.0, 0.0, 1.0);
final_color = mix(final_color, vec4(1.0, 1.0, 1.0, 1.0), foamFactor0);
final_color = mix(final_color, vec4(1.0, 1.0, 1.0, 1.0), foamFactor1);
// Fog
float fogFactor = computeFogFactor( g_FogDensity, gl_FogFragCoord );
water_color = final_color;
 // Sun spots
float cos_spec = clamp(dot(reflect_vec, g_SunDir), 0.0, 1.0);
float sun_spot = pow(cos_spec, g_Shineness);
water_color.xyz += g_SunColor * sun_spot;
gl_FragColor = water_color;
```

Box 1: The implementation detail.

Table 1: Timing per frame for various examples in millisecond.

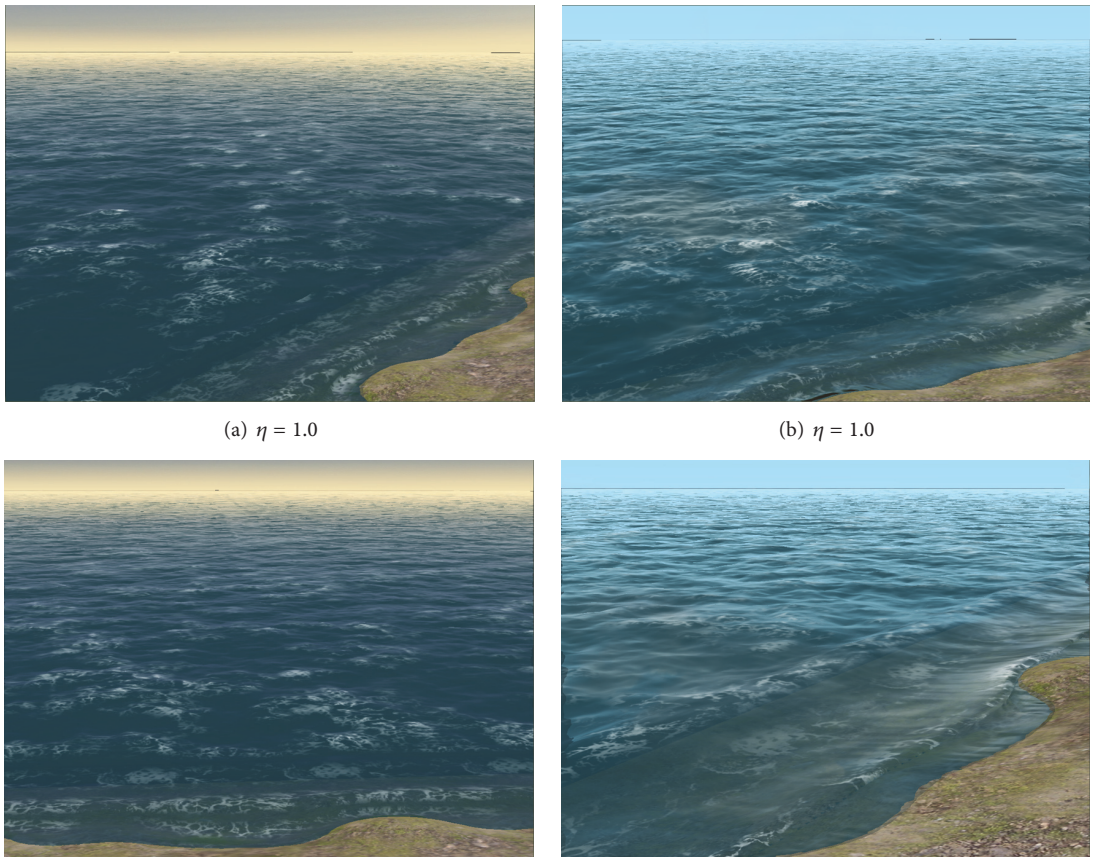
Grid size	Computation grid size	Model	Rendering	Total
512^2	128^2	6.56	5.06	12.06
	256^2	21.67	13.10	36.85
	512^2	115.89	18.56	134.58
1024^2	128^2	6.58	5.33	13.83
	256^2	22.36	14.07	38.02
	512^2	116.23	21.99	138.69

Table 2: A comparison of the difference between Luo's method, Ojeda's method, and our method.

	Ojeda's method		Luo's method	Our method
Computation platform	CPU	GPU	GPU	CPU
Computation grid size	128^2	128^2	250^2	128^2
Water scale	Small	Small	Middle	Large
Averaged timing per frame	410.88 ms	5.28 ms	16.67 ms	12.06 ms

the water model. We can also handle any underlying terrain since the model is computed in real-time. Furthermore, we also consider the deep water when the depth is getting larger. It can also help to make the coastal scene more real.

One drawback of our method is that the rendering results lack higher details. The geometric based model is not the same as the physically based model, and it cannot simulate the higher details, such as splash and breaking dam. The other



(a) $\eta = 1.0$ (b) $\eta = 1.0$

(c) $\eta = 1.2$ (d) $\eta = 1.2$

Figure 7: The rendering result of coastal water scene with different thin coefficients.

problem is that the waveform will become crossed if we do not limit the thin coefficient. However, our method offers a high rendering speed even on an ordinary graphic hardware. This can allow more consumer-level computers to run the coastal water simulation.

6. Conclusions

In this paper, we present an automatic optimization approach for the rendering of large-scale real-time coastal water. A hybrid coastal water model is estimated for large-scale coastal water according to the depth of water. The deep water is also considered in our hybrid coastal water. Unnecessary grid cells are simplified by our automatic simplification algorithm. Finally, we implement our method on GPU by using shader language. The reflection and foam effect are also added to the water surface to enhance the final rendering result.

There are several ways to further improve the visual appearance of the coastal water. The extraordinary detailed effects like waterfall, breaking dam, and splashes are needed to be complex modeled by using particles systems. The interaction with rigid bodies is also a challenging work in the future.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

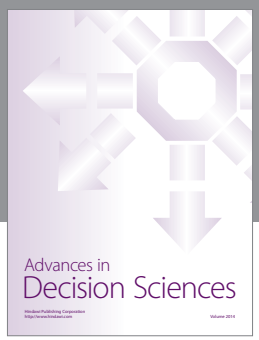
[1] T. Kellomäki, "Water simulation methods for games: a comparison," in *Proceedings of the 16th International Academic MindTrek Conference (MindTrek '12)*, pp. 10–14, Tampere, Finland, October 2012.

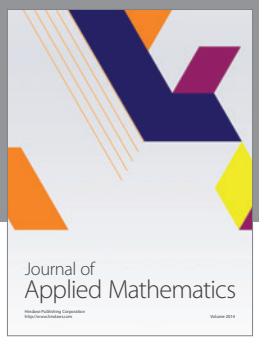
[2] D. P. Playne, K. A. Hawick, and M. G. B. Johnson, "Simulating and Benchmarking the shallow-water fluid dynamical equations on multiple graphical processing units," in *Proceedings of the 12th Australasian Symposium on Parallel and Distributed Computing (AusPDC '14)*, vol. 152, pp. 29–35, Auckland, New Zealand, January 2014.

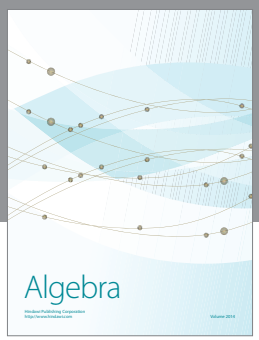
[3] N. Chentanez and M. Müller, "Real-time eulerian water simulation using a restricted tall cell grid," in *Proceedings of the ACM Transactions on Graphics*, pp. 82:1–82:10, New York, NY, USA, 2011.

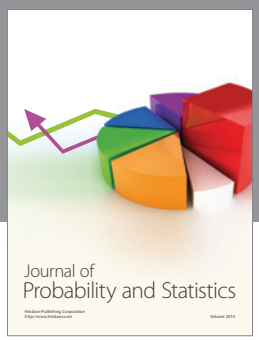
[4] J. Ojeda and A. Susin, "Hybrid particle Lattice Boltzmann shallow water for interactive fluid simulations," in *Proceedings of the 8th International Conference on Computer Graphics Theory and Applications (GRAPP '13)*, pp. 217–226, Barcelona, Spain, 2013.

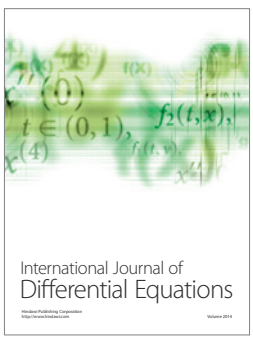
[5] J. Ojeda and A. Susin, "Real-time rendering of enhanced shallow water fluid simulations," http://www.pagines.ma1.upc.edu/~toni/files/hybrid_lbmsw_rend_new.pdf.
[6] I. Abbasov, "Three-dimensional simulation the runup of nonlinear surface gravity waves on shallow coast," *International Journal of Pure and Applied Mathematics*, vol. 93, no. 1, pp. 135–145, 2014.
[7] E. Darles, B. Crespin, and D. Ghazanfarpour, "A particle-based method for large-scale breaking waves simulation," in *Computer Vision and Graphics*, pp. 316–323, Springer, Berlin, Germany, 2010.
[8] M. Chládek and R. Ďurikovič, "Particle-based shallow water simulation for irregular and sparse simulation domains," *Computers & Graphics*, vol. 53, pp. 170–176, 2015.
[9] T. Kellomäki, "Rigid body interaction for large-scale real-time water simulation," *International Journal of Computer Games Technology*, vol. 2014, Article ID 580154, 12 pages, 2014.
[10] A. Fournier and W. T. Reeves, "A simple model of ocean waves," *ACM Siggraph Computer Graphics*, vol. 20, no. 4, pp. 75–84, 1986.
[11] D. S. De Lima, H. Braun, and S. R. Musse, "A model for real time ocean breaking waves animation," in *Proceedings of the 9th Brazilian Symposium on Games and Digital Entertainment (SBGames '10)*, pp. 19–24, Florianopolis, Brazil, November 2010.
[12] M. Luo, G. Gong, and A. El Kamel, "GPU-based real-time virtual reality modeling and simulation of seashore," in *Advances in Robotics and Virtual Reality*, vol. 26 of *Intelligent Systems Reference Library*, pp. 307–332, Springer, Berlin, Germany, 2012.
[13] N. Chentanez and M. Müller, "Real-time simulation of large bodies of water with small scale details," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 197–206, Madrid, Spain, July 2010.
[14] S. Longo, M. Petti, and I. J. Losada, "Turbulence in the swash and surf zones: a review," *Coastal Engineering*, vol. 45, no. 3-4, pp. 129–147, 2002.
[15] K. Fang, Z. Liu, and Z. Zou, "Modelling coastal water waves using a depth-integrated, non-hydrostatic model with shock-capturing ability," *Journal of Hydraulic Research*, vol. 53, no. 1, pp. 119–133, 2015.
[16] E. Bruneton, F. Neyret, and N. Holzschuch, "Real−time realistic ocean lighting using seamless transitions from geometry to BRDF," in *Computer Graphics Forum*, pp. 487–496, 2010.
[17] Y. Yin, H.-X. Ren, and X.-W. Liu, "Real-time wave simulation based on wave spectrum used in marine simulator," in *Proceedings of the Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing (ICSC '08)*, pp. 225–229, October 2008.
[18] Y. Li, Y. Jin, Y. Yin, and H. Shen, "Simulation of shallow-water waves in coastal region for marine simulator," in *Proceedings of the 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI '08)*, p. 15, December 2008.
[19] C. Li, D. Pickup, T. Saunders et al., "Water surface modeling from a single viewpoint video," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 7, pp. 1242–1251, 2013.
[20] S. Wang, F. Kang, and D. Wang, "Ocean wave real-time simulation based on adaptive fusion," in *Proceedings of the 32nd Chinese Control Conference (CCC '13)*, pp. 8557–8560, IEEE, Xi'an, China, July 2013.
[21] S. Wang, F. Kang, and J. Xu, "Modeling and visualization of curved waves near seashore," *Journal of System Simulation*, vol. 26, no. 10, pp. 2395–2399, 2014.
[22] Y. Yin, Y.-C. Jin, and H.-X. Ren, "Wave simulation of visual system in marine simulator based on wave spectrums," in *Proceedings of the International Conference on Marine Simulation and Ship Maneuverability (MARSIM '03)*, Kanazawa, Japan, August 2003.
[23] D. Hinsinger, F. Neyret, and M.-P. Cani, "Interactive animation of ocean waves," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 161–166, San Antonio, Tex, USA, July 2002.
[24] X. Cui, J. Yi-Cheng, and L. Xiu-Wen, "Real-time ocean wave in multi-channel marine simulator," in *Proceedings of the ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI '04)*, pp. 332–335, June 2004.
[25] C. Johanson and C. Lejdfors, *Real-time water rendering [M.S. thesis in computer graphics]*, Lund University, Lund, Sweden, 2004.

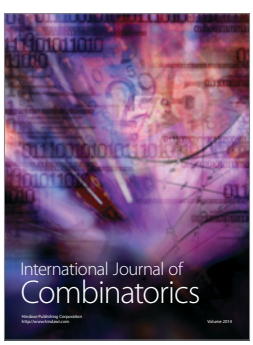


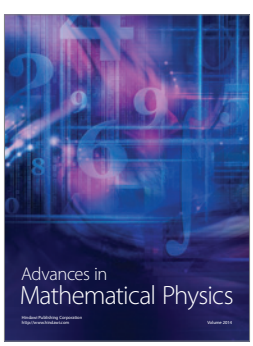


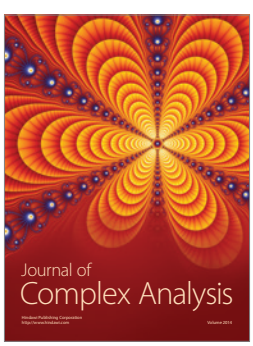


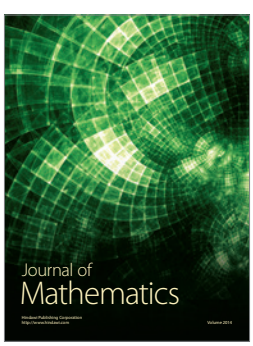


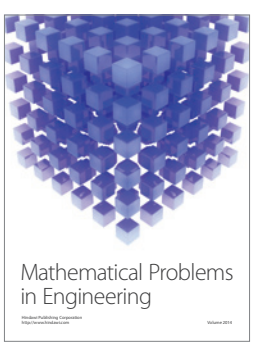


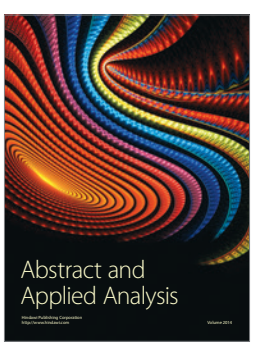


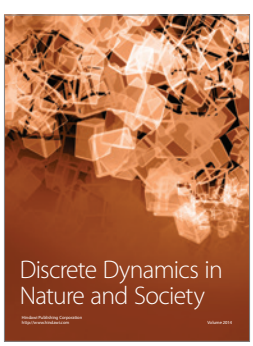


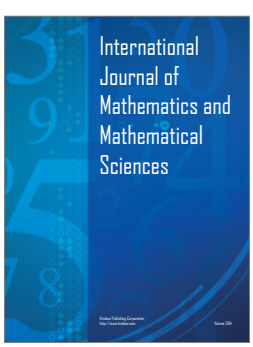


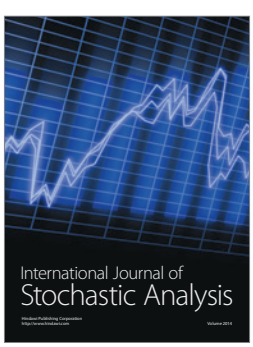












Hindawi

Submit your manuscripts at
http://www.hindawi.com