Efficient View-Dependent Modeling and Rendering of Large-Scale Ocean Wave base on Digital Earth

Li Sunjun, Yang Bing, and Wu Lingda

Multimedia R&D Center, National University of Defense Technology, Changsha, P.R. China 410073

Summary
The realistic modeling and rendering of ocean surface is one of hotspot and difficult problem of computer graphic. In this paper, considering earth curvature, an efficient view-dependent modeling and rendering method of large-scale ocean wave is presented. First, considering the effect of earth curvature in detail, the authors set up the ocean wave model based on digital earth and the knowledge of ocean. It reduces the computation obviously and resolves the "compress phenomenon" at high latitude. Then, relying on a procedural ocean wave model, the method restricts computations to the visible part of the ocean surface with screen-subdivision algorithm, adapts the geometric resolution to the viewing distance: high resolution in near area and low resolution in far area. This yields real-time performances, even when the camera moves. The "border slit" on screen can be removed by the method based on "Clamp Sphere". Finally, various ocean waves under different distance and different viewpoints are realistically view-dependent rendered. Experiments have been implemented on a common PC with standard graphics hardware. Experimental results show that the user can interactively fly over a large-scale earth animated ocean.

Key words:
Ocean wave, View-dependent modeling and rendering, LOD, Screen-subdivision algorithm.

1. Introduction

The realistic modeling and rendering of ocean surface is one of hotspot and difficult problem of computer graphic[1]. It is wide used in virtual battle field, navigate drilling, game, fish culture, forecast of ocean, and so on. Most of game and drilling system is base on or aimed at the flat of digital globe. So modeling and rendering of ocean surface based on digital globe (MROSDG) is urgent.

Most of the recent ocean wave simulation is based on plane and is in a limited region. They didn't consider the factor of earth curvature and can't used on digital earth flat. Moreover, ocean wave animations were created using a spectral approach. The computation cost only allows crude resolution when interactive rates are required. Moreover, enabling the user to interactively fly over the ocean is impossible, since avoiding obvious repeat would require the use of an extremely large simulated region[2].

Earth surface is different from the plane, at rest, it is in 3D space. Modeling of earth ocean surface is puzzled by the curvature and great computation cost. The main difficulty include: the modeling of earth ocean wave, the effect of earth curvature, the changing of the visual of globe surface with the movement of viewpoint, the creating of continuous and dynamic resolution grid, the computation of ocean color with the movement of viewpoint and so on. At the same time, ocean is not at rest, it is not only dynamic in space, but also in time, moreover interactive dynamic in space and time. Ocean surface is massive and 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. So, MROSDG is very difficult.

In this paper, a new ocean wave modeling and rendering method considering earth shapes is proposed, its kernel is the procedural modeling of globe ocean waves and the screen-subdivision algorithm. The authors first set up the ocean wave model according to the knowledge of ocean and the characters of earth surface, sample the visual spherical surface with screen-subdivision algorithm. The sample is adapted to the current viewpoint, is continuous and dynamic resolution. Relying on a procedural ocean wave model, it restricts computations to the visible part of the ocean surface. This yields interactive rates, even when the camera moves under the user control. Finally, various ocean waves under different distance and different viewpoints are realistically view-dependent rendered. Experiments have been implemented on a common PC with standard graphics hardware. Experimental results show that the user can interactively fly over a large-scale globe animated ocean.

The remainder of this paper develops as follows: Section 2 reviews the existing models for simulating ocean waves. Then we present our wave model of globe surface in Section 3. Screen-subdivision adaptive grid is detailed in Section 4. We show implementation and result in Section 5, conclusion and future work in Section 6.

2. Related Work

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

Manuscript received May 5, 2007.
Manuscript revised May 20, 2007.

2.1 Based on geometrical models

Blinn[3] presents the bump mapping technique developed in 1978. This method allows realistic rough textured surfaces to be obtained by perturbing the surface normal. In 1980, Fishman and Schachter[4] introduce the "height field" technique and later modified by Max[5]. Fournier and Reeves[6] introduce a scheme for ocean waves simulation including wave refraction and other wave effects in 1986. Their proposal was based on the Gerstner-Rankine model, proposed in oceanography long ago [7, 8]. Roughly speaking, this model describes that particles of water describe circular or elliptical stationary orbits.

Generally speaking, the above approaches are very simple, needn't immense computation, and near real-time, but they need to adjust the parameters in reason and the scenes are less realistic.

2.2 Based on physical models

Because the Navier-Stodes equations (NSE) describe completely the motion of a fluid at any point within a flow at any instant of time, it is often used to simulate fluid phenomena accurately. N. Foster and J.X. Chen[9-11] have applied the NSE to simulate ocean wave in 1995-2000. In 2001, Premoze and Ashikhmin[12] introduced a method for wave generation on water surfaces using a physically-based approach and described a non-real-time light transport approach for computing complex lighting effects of ocean.

Those approaches are so complex that it needs so immense computation that it can't simulate the ocean wave in real-time. It can be used to create the realistic picture offline.

2.3 Based on wave spectrum

Spectral approaches are first introduced in CG by Mastin et al[13]. The basic idea is to produce an height field having the same spectrum as the ocean surface[2]. In 2001, Jensen[14] and Tessendorf[15] separately describe statistical and experiential models of ocean water, simulate the ocean surface with the synthesized height map generated by a summation of sine waves using the FFT method.

For the approach, a pre-defined region of the ocean surface is simulated, whatever the camera position. Since the visible surface area is large and a high resolution is needed in regions close to the camera, this requires very large meshes, yielding costly simulations. These methods can only be adapted to real-time for very crude resolutions[2].

2.4 Based on Perlin noise

Perlin[16] presents the Perlin noise function developed in 1985. Perlin noise is a continuous noise. This method can generates realistic high map of ocean surface by using the Perlin noise function. Perlin[16] used noise synthesis approach to simulate the appearance of the ocean surface seen from a distance. In 2004, Johanson[17] just used Perlin noise synthesis approach to simulate ocean waves, the scene is very realistic.

This way can be implemented very simple and have a more photorealistic ocean. Since it's complex of computation, it is used to simulate small region ocean surface.

2.5 Based on Particle System

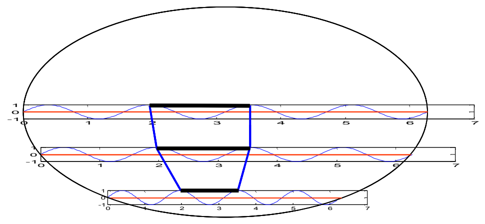
The particle systems were firstly introduced by Reeves [18] (see also [19]) to describe natural phenomena. In the water simulation schemes, they were generally applied to model the foam and the spray generated by wave breaking and collisions with obstacles [6, 20]. The characteristics and shape of the particles change with time under the action of physical or stochastic models.

If the model and computation of particle system of foam and spray are very complex, those ways are fit to animation and not to real-time. If very simple, the scene is less realistic.

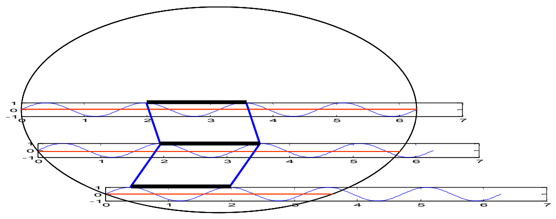
Besides the above five types of ocean waves models, Stefan[21] presented a procedural model for breaking ocean waves. Wang [22] presented a new approach using cellular automata and put forward a new evolution rule of cell is to mimic the motion of ocean waves.

3. Ocean wave model of spherical surface

Our main contribution is the globe ocean wave model considering the curvature of earth and computation cost. Modeling of earth ocean surface is puzzled by the curvature and great computation cost. Earth surface is different from the plane, at rest, it is in 3D space. It should models with three variables: x, y, z. So the quantity of computation of the points on globe is 3^N and the cost of computation is very high. In order to improve the efficiency of implement, the quantity of computation must be decreased. Supposing the radius of earth is constant or fixed value, we can model the globe ocean surface with two variables: longitude, latitude. This yields that the quantity of computing the points on globe is 2^N. It reduces the quantity of computation obviously and the cost of computation is comparatively low. At the same time, the method modeling earth ocean wave with two variables brings another problem: the "compress phenomenon" at the high latitude, there are more wave crests at the high latitude for compressing, see Fig.1-a. It is very obvious at the north and south poles: there may be a lot of points if it computed by the model. In fact, north pole or south pole is only a point. So we must resolve the "compress phenomenon". The idea is that the same surface length has the same number of wave crest. Adding a parameter $\cos\beta$ (β: the latitude of point) to the model, the circle length of different latitude and the coordinate of vertex (α,β) can be revised, such as (α,β) to $(\alpha,\beta\cdot\cos\beta)$, this skill can resolves the "compress phenomenon", see Fig.1-b. The detail is as follow:



a



b

Fig.1 Globe ocean surface (unwrap to plane)

The noise height model of vertex (α,β) is:

$$fnoise(\alpha,\beta) = f(\alpha\bullet\cos\beta,\beta) \tag{1}$$

Here:

α, longitude of vertex

β, latitude of vertex

$f(\alpha\bullet\cos\beta,\beta)$, the noise height of vertex (α,β)

$\cos\beta$, revise coefficient

Then the actual height of vertex (α,β):

$$\begin{aligned} p_{heightfield}(\alpha,\beta) &= p_{earth}(\alpha,\beta) \\ &+ f(\alpha\bullet\cos\beta,\beta)\bullet N_{earth} \end{aligned} \tag{2}$$

Here:

$-\pi\le\alpha\le\pi, -\pi/2\le\beta\le\pi/2$

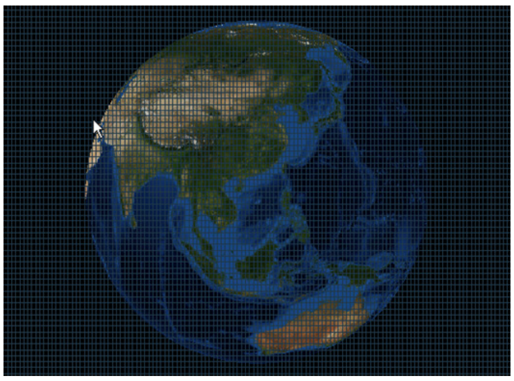
$p_{heightfield}$, the actual height of vertex (α,β) in real-time

$p_{earth}(\alpha,\beta)$, the rest height of vertex (α,β) at rest

N_{earth}, the normal of tangent plane of vertex (α,β)

The earth ocean mesh can be created using Eq.(2). At equator, $\beta=0$, $(\alpha,0)$ is a set of point. The noise height $f(\alpha,0)$ can be computed using the Eq.(1). At the north and south poles, $\beta=\pm\pi/2$, $(\alpha\bullet\cos\beta,\beta)$ is only a point $(0,\beta)$. The noise height $f(0,\beta)$ can be computed with the Eq.(1). We can compute the other points between the equator and the pole with the same principle. The value of $p_{heightfield}(\alpha,\beta)$ is the distance from point (α,β) to the center of globe. With the Changing of $p_{heightfield}(\alpha,\beta)$, we can create the dynamic globe ocean wave.

4. Screen-subdivision adaptive grid



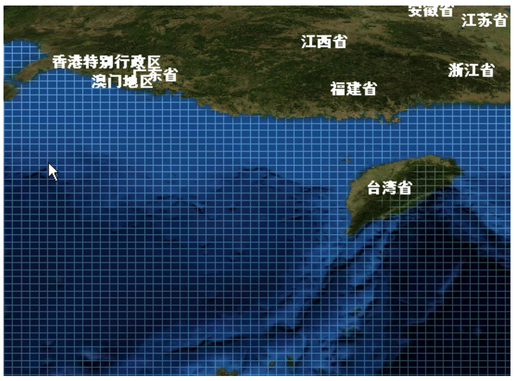


Fig.2 Snapshot of globe mesh

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 [2, 17, 23], but the

LOD is not oversimplified or can't used to ocean simulating of digital earth. If we want to implement the simulation of ocean surface on the digital globe flat, we must use LOD algorithm to reduce the numbers of triangle patches.

Our main contribution is the adaptive scheme of globe mesh: Instead of simulating a regularly sampled, predetermined region of the ocean surface, we adapt the mesh sampling to the projected size of each globe surface element, the main idea is to generate the globe mesh representing the globe ocean surface such that every element covers, at rest, the same area on the screen. This is done by subdividing the screen into a grid of quads, which are back-projected on the globe modeling the ocean surface at rest. The resulting mesh points provide the particle locations at which the procedural animation model of Eq.(2) is evaluated (see Fig.2).

The implement is done by setting up a plane in view space, which is upright with the direction of viewer, subdividing the plane into a grid of quads, then connect the viewpoint and the point of plane mesh, compute intersections the line intersects the earth modeling the ocean surface at rest, the resulting intersections near viewpoint provide the particle which is back-projected on the digital earth modeling the ocean surface at rest. The resulting mesh points constitute the visual part globe mesh.

4.1 Coordinate translation

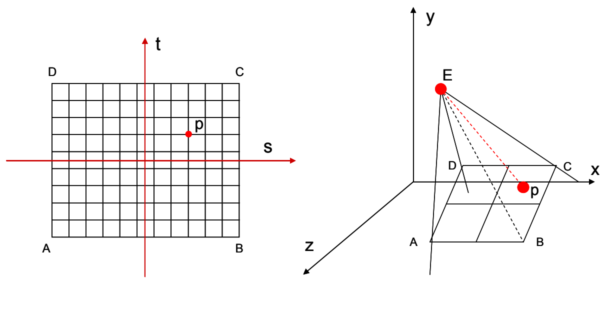


Fig.3 Coordinate Translation

First, we set up a screen coordinate system (SCS) (see Fig.3), set a quadrangle $ABCD$ in screen coordinate system, here: $s \in (-1.0,1.0)$. $t \in (-1.0,1.0)$. Then, definite translation matrixes and points used in this paper:

M_{View} : view matrix of camera,

$M_{Perspective}$: perspective matrix,

$M_{\Pr oj} = M_{View} \cdot M_{Perspective}$: projector matrix of camera,

$M_{Inv\Pr oj} = [M_{\Pr oj}]^{-1}$: the inverse matrix of projector matrix of camera,

$p_s(s,t,z_s)$: a point in screen space, $-1 \le z_s \le 1$

$p_w(x,y,z)$: a point in world space.

A point is transformed from world-space to screen-space by transforming it through the view and perspective matrices ($M_{View} \cdot M_{Perspetive}$) of the camera. This is done for all geometry that is defined in world-space. If that transform is inverted, we obtain the following:

$$p_s(s,t,z_s) = M_{\Pr oj} \cdot p_w(x,y,z) \qquad (3)$$

$$p_w(x,y,z) = M_{Inv\Pr oj} \cdot p_s(s,t,z_s) \qquad (4)$$

The translation matrixes $M_{\Pr oj}$ and $M_{Inv\Pr oj}$ are not dependent of the sampling points, but only dependent of viewpoint, the direction of viewer and reference point. Therefore, the translation matrixes $M_{\Pr oj}$ and $M_{Inv\Pr oj}$ need computed only when the viewpoint, the direction of viewer or reference point is changed.

4.2 Multi-resolution adaptive mesh of ocean surface of digital earth

Eq.(3) and Eq.(4) show how a point is transformed between screen space and world space. If we consider the z-coordinate (the depth) of each point in screen space to be undefined, there will for each resulting point be a line in which the source point could be located (in world space). This is because the space of the source point has a higher rank. If the resulting line is intersected with the earth surface we will obtain the world-space position.

This can be done as follow:

First, we compute the two corresponding points (viewpoint: $E(x,y,z)$, far-point: $C(x,y,z)$) in world space of a point in screen space as follow:

$$E(x,y,z) = M_{Inv\Pr oj} \times p(s,t,-1) \qquad (5)$$

$$C(x,y,z) = M_{Inv\Pr oj} \times p(s,t,+1) \qquad (6)$$

Then, the line passing the two pints (viewpoint: $E(x,y,z)$, far-point: $C(x,y,z)$) is computed.

$$\frac{x-C_x}{E_x-C_x} = \frac{y-C_y}{E_y-C_y} = \frac{z-C_z}{E_z-C_z} \qquad (7)$$

Finally, we compute the intersections of line EC and earth surface at rest.

$$\begin{cases} \dfrac{x-C_x}{E_x-C_x} = \dfrac{y-C_y}{E_y-C_y} = \dfrac{z-C_z}{E_z-C_z} \\ x^2+y^2+z^2=R^2 \end{cases} \tag{8}$$

Getting two intersections of each line, the nearer intersection to viewpoint is the right intersection ($P(x,y,z)$). We can get the intersection ($P(\alpha,\beta,r)$) in polar coordinates using coordinate translation.

We can get the height $p_{heightfield}(\alpha,\beta)$ of point $P(\alpha,\beta,r)$ using Eq.(1) and Eq.(2) at time t. The point $P'(x,y,z)$ in world coordinates of point $P(\alpha,\beta,r)$ can be computed using coordinate translation. This yields the globe wave mesh with a set of points as $P'(x,y,z)$.

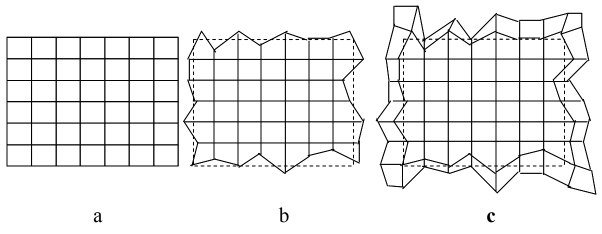


Fig.4 Slit of screen

For the movement and creating globe mesh with points at rest, the actual height of ocean surface may higher or lower than ocean surface at rest. This yields the project of ocean surface in world space can't covers the all screen, as leads the slit in boundary of screen (see Fig.4-b: some border points inside screen (dashed rectangle)).

The detail reason of screen slit is: it is supposed that the height of $f(\alpha \bullet \cos\beta, \beta)$ is between $-h$ and h. There are three circles (outer circle, base circle and inner circle called in this paper) in figure 5 that is a slice of globe. The distance between two adjacent circles is h. So the globe ocean surface is between the outer globe and inner globe. i) When the viewpoint is far, the view scope is the part in $\angle AEB$, see Fig.5-a. ii) When the viewpoint is near enough that we can't view all the globe ocean surface but only a part, such as the part of $\angle C'E'D$ in Fig.5-b, magnifying the part in $\angle C'E'D$, the result display as Fig.5-c. The part in $\angle PEM$ can be seen in screen at rest. For the movement of ocean surface and creating globe mesh with points on base globe, the ocean surface creating with the points of arc PM will be in the side of screen and not on the border of screen sometime (see Fig.5-c: read line). This yields the "screen slit" in the border of screen.

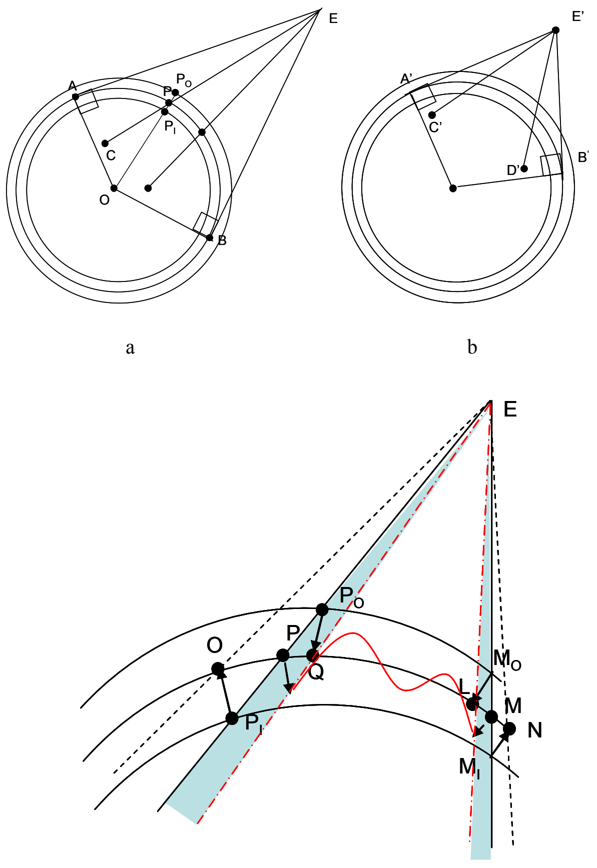


Fig.5 A slice of digital globe passing viewpoint

How to remove the "screen slit"? In this paper, "the border line" in world space is a point in the border line of screen space, such as EP. First, we can compute the screen coordinates of the points of Q, P and O, get the maximum and minimum. The other maximum and minimum of border lines can be computed using the same method. Then, we can get the max and min from these maximum and minimum. Finally, the screen rectangle that is used to generate the globe ocean mesh can be composed of the four points of A (min, min), B (max, min), C (max, max) and D (min, max). Subdividing the screen into a grid of quads, which are back-projected on the earth modeling the ocean surface at rest, we create the dynamic ocean wave with the resulting mesh points. This can remove the "screen slit", see Fig.4-c (all border points outside screen (dashed rectangle)).

4.3 View-dependent rendering of visible ocean surface

It would render every pixel points using the screen subdivision. Since earth and ocean is just a part of screen, if only rendering the visible part of ocean surface, the efficiency would be improved obviously. The rendering part can be decided by computing the visible part of earth in screen, patch subdivision and the judging of ocean and land.

4.3.1 Judging of the visible part of digital earth.

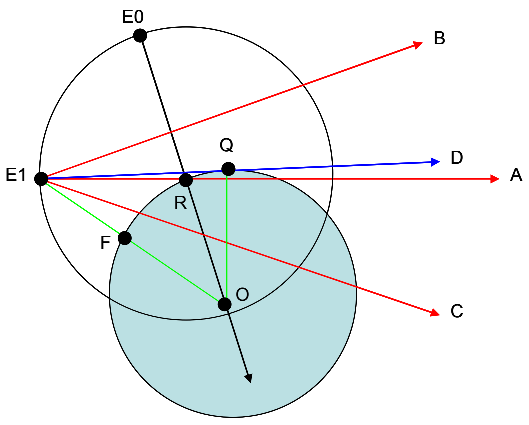


Fig.6 Slice of digital earth

Fig.6 is a slice of digital earth passing eye-point, reference point and earth center.

Here:

E_1 the current eye-point.

R the reference point

$\angle BE_1C$ the visible angle, 45 degree

E_1Q the tangency line

It can be got based the digital earth:

$$D_{OR}=D_{OQ}=D_{OF},\ D_{OE_1},\ D_{E_1F},\ D_{E_1R}=D_{E_0R},\ D_{E_1F},\ \angle E_0RE_1=Tile.$$

It can be get based the geometry:

$$\angle QE_1R=\angle QE_1O-\angle RE_1O$$

Because,

$$\angle QE_1O=\arcsin\frac{D_{OQ}}{D_{E_1O}}$$

$$\angle RE_1O=\arcsin\frac{D_{OR}*\sin\angle E_1RO}{D_{OE_1}}$$

$$\angle E_1RO=\pi-\angle E_0RE_1$$

Therefore,

$$\begin{aligned}\angle QE_1R&=\arcsin\frac{D_{OQ}}{D_{E_1O}}-\arcsin\frac{D_{OR}*\sin\angle E_1RO}{D_{OE_1}}\\ \Rightarrow\angle QE_1R&=\arcsin\frac{D_{OQ}}{D_{E_1O}}-\arcsin\frac{D_{OR}*\sin Tile}{D_{E_1O}}\\ \Rightarrow\angle QE_1R&=\arcsin\frac{D_{OQ}}{D_{E_1F}+D_{OF}}-\arcsin\frac{D_{OR}\sin Tile}{D_{E_1F}+D_{OF}}\end{aligned}\tag{9}$$

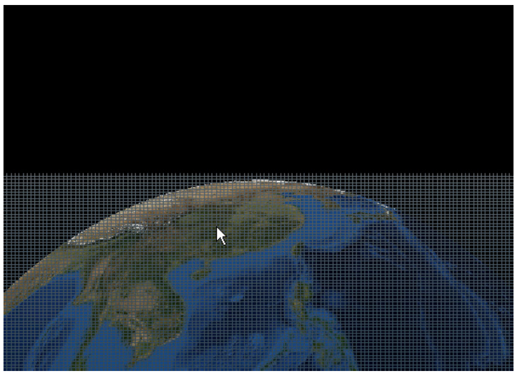


Fig.7 Computing the visible part of digital earth

$\angle QE_1R$ can be get by Eq.(9). If $\angle QE_1R\ge 22.5*\pi/180$, the screen is covered by digital earth, or there is a $0\sim\frac{22.5*\pi/180-\angle QE_1R}{45*\pi/180}$ part is not covered in the upper of screen (see Fig.7). So, the part needn't rendered, the computation and rendering can be reduced about $\frac{22.5*\pi/180-\angle QE_1R}{45*\pi/180}$.

4.3.2 Patch subdivision.

By dividing the grid into a smaller number of patches, it is possible to prevent unnecessary processing to occur. Fig.8 looks very similar to approaches that could be used for LOD-based schemes. The main difference is that the division of the grid is done in screen-space, which among other things means that the rendered patches depend on the camera position.

It can reduce overdraw by Patch subdivision. The red region is the ocean lines of (e120n23e). The data come form NOAA, the interval between points is 30 meters, round error of the point is about 20 meters. Now, the red region can be rendered by only rendering the gray grid patches.

The proposed solution is as follows:

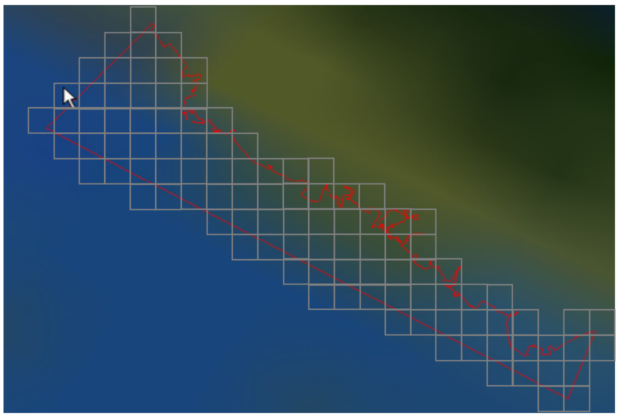
(i) Subdivide the screen grid into patches

(ii) For each screen grid-point, determine whether the point is ocean surface of (e120, n23) and could be visible.

(iii) For each screen patch, determine whether the patch could be visible: as long as a point of the four points of the patch can be visible, the patch is visible.
(iv) Compute the dynamic position of each screen grid-point using the method of 3, 4.1 and 4.2 segment.
(v) Render the visible screen patches.

Using the method, the patches that are visible can be get (see Fig.8-a), the rendering of the patches can be seen in Fig.8-b (the points that are not visible are clarity).

Patch subdivision will save both vertex and pixel bandwidths, decrease the computation, and improve the rendering rate. Figure 8 illuminates that the approach is feasible.


a

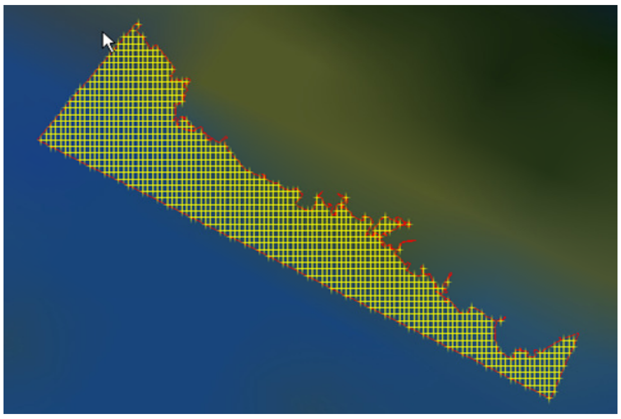

b

Fig.8 Patch subdivision.
Red line is the ocean line of (e120n23e). The patches in gray grids are visible part of the ocean surface of (e120n23e).

4.3.3 Using scissors

Scissors could also be used to limit the ocean surface. The approach is simple. Set the distance between the eye-point and the near scissor plane as 0, and the distance between the eye-point and the far scissor plane (it is dynamically changed) as the distance between eye-point and tangency-point. Thus, the objects that behind the d far scissor plane will be deleted.

5. Implementation and result

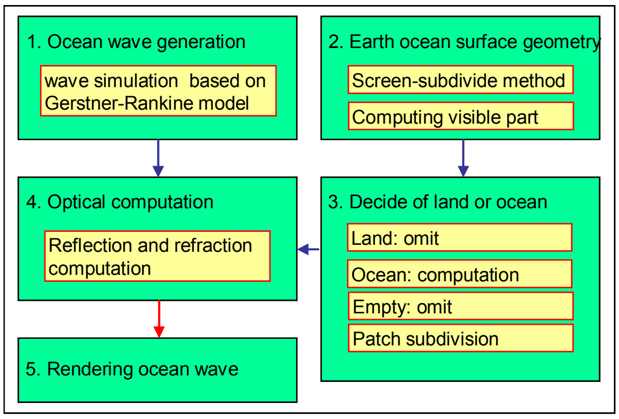


Fig.9 Overview of ocean wave simulation system based on digital earth

Our ocean wave simulation system based on digital globe is divided into five stages (See Fig. 9): ocean wave generation, surface tessellation, decide of land or ocean, optical simulation, and water surface rendering. Except the first stage (wave generation), they are all implemented in GPU.

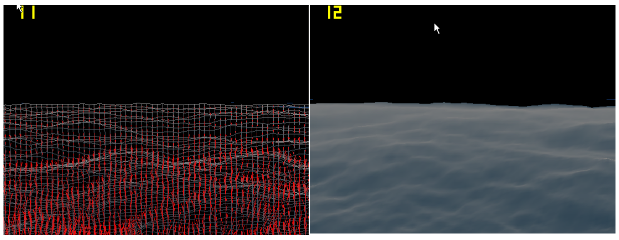


Fig.10. Snapshot of mesh, normal and effect of globe ocean

We made experiments on microcomputer. Its configuration is PIV3.0, 512M memory and 256M display card (RADEON X600). The development environment is Windows XP system, Visual C++2005 and DirectX 9.0. Here we not only emphasize the fast rendering with many complex objects to create an immersing virtual ocean environment, but also the realistic images. In addition to using an optimized mesh with continuous LOD, we apply the Fresnel function to compute the color of per vertex in real-time. Using the computing method of vertex color can get realistic ocean scene whatever the viewpoint and the direction of view is. The results illustrated in Fig.10: more than 10 fps are obtained with a 512 × 512 screen-mesh resolution. Real-time images can still be obtained more than 45 fps (Fig.11) with 64×64.

We improved the method of [2] in that we recreate a globe ocean surface mesh and can used for the digital globe flat. Our method is more useful than [17]: the creating ocean surface is unbound, we can simulate and navigate not only in a local ocean surface but also in the all digital globe from near to far.

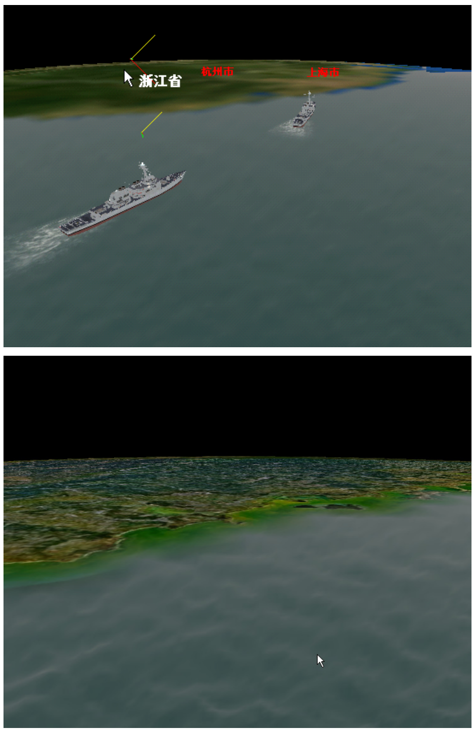


Fig.11 Snapshot of integrate scene of land and ocean

6. Conclusion and future work

First, the paper presents an ocean model considering the curvature of earth based on the theory of oceanography, computer graphics and analytic geometry. Then, it deduces the ration expression of the height map of different longitude, latitude and viewpoint, resoles the "compress phenomenon" caused by the change of latitude, creates the ocean mesh with an adaptive screen-subdivision algorithm. Finally, realistic globe and local ocean scenes are display, as is not mentioned in other papers.

Comparing to others, the work of this paper has some characteristic:

(i) The main target of this paper that is to simulate all or local globe ocean wave scene is different form other papers that are to simulated partial or local plane ocean wave scene. It discusses and analyses the problems in the MROSDG, then presents a feasible model.

(ii) The model can simulate and render some scenes that other ocean models can't simulate, such as the integrative scene of land and ocean.

(iii) It resolved the "compress phenomenon" caused by the changing of latitude.

(iv) The adaptive model presented in this paper can render unbound ocean wave and interactive flyover on the DGF.

(v) The method restricts computations to the visible part of the ocean surface, the generated mesh density transition is continuous and an adequate resolution being maintained everywhere in the computed ocean image. The method provides a flexible control of the quality/cost ratio by adjusting the mesh density and the number of simulated noises.

(vi) Moreover, our method works at no extra cost whatever the eye-point motion or not, as is very useful for simulator training and scientific research.

Future research include: Improving the ocean model and using the GPU technique to accelerate rendering rate, modeling and rendering the curve and break of the ocean wave, modeling and rendering the reflection and refraction of ocean wave, simulating the integrated scenes including land, ocean and sky.

Refernce:

[1] Iglesias, A., Computer graphics for water modeling and rendering: a survey[J]. Future Generation Computer Systems, 2004. 20: p. 1355-1374.
[2] D, H., N. F, and C.M. P, Interactive Animation of Ocean Waves[C]. Proceedings of ACM SIGGRAPH Symposium on Computer Animation, 2002: p. 161-166.
[3] Blinn, J.F., Simulation of wrinkled surfaces[C/J]. in: Proceedings of SIGGRAPH'78, Comput. Graph., 1978. 12 (3): p. 286-292.
[4] B.Fishman and B.Schachter, Computer display of height fields[J]. Comput. Graph., 1980. 5: p. 53-60.
[5] Max, N.L., Vectorized procedural models for natural terrain: waves and islands in the sunset[C/J]. SIGGRAPH'81, Comput. Graph., 1981. 15 (3): p. 317-324.
[6] Fournier, A. and W.T. Reeves, A simple model of ocean waves[C/J]. Proc. SIGGRAPH'86, computer graph., 1986. 20(4): p. 75-84.
[7] Gerstner, F.J., Theorie der wellen, Ann. der Physik 32. 1809: p. 412-440.
[8] Rankine, W.J.W., On the exact form of waves near the surfaces of deep water. Phil. Trans. R. Soc. A, 1863. 153 (4): p. 127-138.
[9] Chen, J.X. and N.V. Lobo, Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations[J]. Graph. Models Image Process, 1995. 57 (2): p. 107-116.
[10] Foster, N. and D. Metaxas, Modeling water for computer animation[J]. Commun. ACM, 2000. 43(7): p. 60-67.
[11] Foster, N., Realistic Animation of Liquids[J]. Graphical models and image processing, 1996. 58(5): p. 471-483.
[12] Premoze, S. and M. Ashikhmin, Rendering Natural Waters[J]. Computer Graphics Forum, 2001. 20(4): p. 189-200.

[13] MASTIN, G.A., P.A. WATTERBERG, and J.F. MAREDA, Fourier synthesis of ocean scenes[J]. IEEE Computer Graphics and Applications, 1987. 7, 3 (Mar.): p. 16-23.
[14] Jensen, Deep-Water Animation and Rendering[EB/OL]. http://www.gamasutra.com/gdce/Jensen/Jensen_01.htm, 2001.
[15] Tessendorf., Simulating Ocean Water[EB/OL]. http://home.gte.net/tssndrf/index.html.2001, 2001.
[16] Perlin, k., An Image Synthesizer[C]. SGGRAPH 85, San Francisco, 1985: p. 287~296.
[17] Johanson, C., Real-time water rendering:[Master dissertation]. Sweden: Department of Computer Science, Lund University, 2004.
[18] REEVES, W.T. and L. Ltd, Particle Systemsm A Technique for Modeling a Class of Fuzzy Objects. ACM Transactions on Graphics, 1983. 20(2): p. 91-108.
[19] Reeves, W.T. and R. Blau, Approximate and probabilistic algorithms for shading and rendering structured particle systems[C/J]. Proceedings of SIGGRAPH'85, Comput. Graph., 1985. 19 (3): p. 313-322.
[20] Peachey, D.R., Modeling waves and surf[C/J]. Proceedings of SIGGRAPH'86, Comput. Graph, 1986. 20 (4): p. 65-74.
[21] Jeschke, S., H. Birkholz, and H. Schmann, A Procedural Model for Interactive Animation of Breaking Ocean Waves[C]. WSCG'2003, February 3-7, 2003, Plzen, Czech Republic., 2003.
[22] Wang, C., Z. Wang, and J. Jin, Real-time Simulation of Ocean Wave Based on Cellular Automata[C]. Proc.Of CAD/Graphics'2003,Macao,China, 2003.
[23] INc, C., OpenSea Maritime Module User's Guide. CSI Inc, 1997.

Li Sujun, born in 1972. He is currently a doctor candidate in Multimedia R&D Center, National University of Defense Technology. His research interests include computer graphics, virtual reality and computer vision.

Yang Bing, born in 1974. Currently a associate professor and doctor candidate in Multimedia R&D Center, National University of Defense Technology, his research interests include virtual reality and computer graphics.

Wu Lingda , born in 1962. Ph.D, professor and doctor supervisor in Multimedia R&D Center, National University of Defense Technology. Her main research interests are virtual reality and multimedia technology.