Dynamic Grid Sea Surface Simulation Using Tessellation

Qing-yun MAO\textsuperscript{1,2,*}, Yao-jie CHEN\textsuperscript{1,2} and Xin YUAN\textsuperscript{1,2}

\textsuperscript{1}Wuhan University of Science and Technology, Wuhan, China
\textsuperscript{2}Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430065, China
\textsuperscript{*}Corresponding author

Keywords: Dynamic grid, Tessellation, Perlin noise, Sea surface simulation.

Abstract. In order to reduce the resource consumption and enhance the effect in the simulation of large-scale ocean scenes, a dynamic grid algorithm based on viewpoint is proposed to remove unnecessary vertices outside the field of view. At the same time, the GPU hardware accelerated Tessellation is used to adaptively subdivide the sea surface grid inside the GPU. Refine the sea surface grid to meet the needs of high-precision grids near the field of view and low-precision grids far away from the field of view. Finally, using Perlin noise texture to simulate the natural sea surface. Experiments show that the algorithm improves the efficiency of generating the sea surface grid and improves the rendering frame rate while exhibiting realistic ocean visual effects.

Introduction

Natural phenomena like fluid motion and water surface are ubiquitous in our daily environment. Presenting natural phenomena more realistically and naturally is one of the most important aspects of improving virtual reality scenarios, especially in various fields related to computer graphics applications. However, the appearance of many natural phenomena like fluid motion and water surface is still a problem in computer graphics.

The research on sea surface simulation has been carried out a lot, in the direction of grid model, wave waveform, calculation acceleration and so on. For the sea surface grid model, the concentric circle grid model \cite{1-3} is similar to the common Clipmap algorithm, its grid regular, the drawing frame rate is high and constant, but because the detail control is simple, when the viewpoint moves up or down, it will cause the sea surface drawing details to be too low or too high. Bo Li \cite{4} and Vladimir Belyaev \cite{5} both give implementation of camera-based grid, and determine discrete LOD by horizontal distance. An adaptive near-surface mesh optimization method \cite{6} is proposed to simulate a large-scale near-surface scene. Varela \cite{7} developed a discretization of the wave spectrum that obtained a sea state statistically more equivalent to the original. Su \cite{8} used CUDA parallel computing to improve the calculation speed, and applied GPU programming technology to achieve rapid rendering of the marine environment. Maheshya et al\cite{9} introduces a spectral wave model based on the Phillips spectrum and render it by a hybrid method focusing on user perception.

In previous studies, GPU-based Tessellation technology was still not widely used. In this paper, a dynamic mesh model based on viewpoint is used to effectively update the entire grid in the movement of the viewpoint and draw the basic grid of the sea surface. At the same time, using GPU Tessellation technology, the sea surface grid is subjected to secondary processing according to the LOD strategy, and the fineness of the sea surface grid is increased, thereby enhancing the effect of sea surface drawing.

Dynamic Sea Surface Grid Algorithms

Water surface rendering requires real-time, realistic reproduction of the three-dimensional surface of the waterscape. According to the actual situation, the close-up details are clear and the foreground details are blurred. At the same time, due to the limitation of the range of the angle of view, the scene behind the angle of view cannot be observed. Therefore, this article will use the
dynamic grid algorithm to generate a dynamic grid based on viewpoints, eliminating vertices outside the camera’s perspective.

Define the XOZ plane as the standard sea surface. To achieve a view-dependent grid, this article creates a camera and a camera-based frustum. The frustum will track the camera movement. The intersection of the frustum and the XOZ plane is used to constrain the grid of the ocean surface so that only a portion of the visible water is present.

\[ Fru = Camera(1 + \sigma |N \cdot Dir_F|) \]

(1)

Where \( \sigma \) is the base magnification factor, \( N \) is the normal vector of the XOZ plane, and \( Dir_F \) is the direction vector of the camera.

The projection grid idea is used to calculate the four boundary points of the intersection of the frustum and the XOZ plane. The LOD method is used in calculating the vertices of the grid. The position of the vertex \( V_{ij}(0 < i \leq P, 0 < j \leq Q) \) in the grid is obtained by the interpolation of the positions of the four boundary points based on the weights, the weights and the vertices and the viewpoints. The distance is related, as shown in Figure 3.

\[
\begin{align*}
V_{ij} &= (1 - (i - 1)d_i)(1 - (j - 1)d_j)w_A V_A \\
      &+ (1 - (i - 1)d_i)(j - 1)d_jw_B V_B \\
      &+ (i - 1)d_i(j - 1)d_jw_C V_C \\
      &+ (i - 1)d_i(1 - (j - 1)d_j)w_D V_D
\end{align*}
\]
Where \( d_i = 1/(P-1) \), \( d_j = 1/(Q-1) \) represents the grid resolution, and the weights of the four vertices are calculated as follows:

\[
\begin{align*}
    w_A &= \frac{1}{w_{\text{dis}}(V_{\text{cam}}, V_A)} \\
    w_B &= \frac{1}{w_{\text{dis}}(V_{\text{cam}}, V_B)} \\
    w_C &= \frac{1}{w_{\text{dis}}(V_{\text{cam}}, V_C)} \\
    w_D &= \frac{1}{w_{\text{dis}}(V_{\text{cam}}, V_D)} \\
    w &= \frac{(1 - (i-1)d_i)(1 - (j-1)d_j)}{w_{\text{dis}}(V_{\text{cam}}, V_A)} + \frac{(1 - (i-1)d_i)(j-1)d_j}{w_{\text{dis}}(V_{\text{cam}}, V_B)} \\
    &+ \frac{(i-1)d_i(j-1)d_j}{w_{\text{dis}}(V_{\text{cam}}, V_C)} + \frac{(i-1)d_i(1 - (j-1)d_j)}{w_{\text{dis}}(V_{\text{cam}}, V_D)}
\end{align*}
\]

(3)

(4)

Tessellation-based Grid Subdivision

The sea surface grid generated by the previous dynamic grid algorithm is rough, and because the distance between the grid at a distance is too large, the details will be lost in this area where the undulations are relatively large, and the grid with too much spacing will affect the normal vector calculation, causing errors in the sea surface lighting effect. Here, in order to further optimize the sea surface grid, Tessellation can be used.

In the OpenGL 4.x version of the rendering pipeline, Tessellation operations can be implemented on graphics with two new shaders, Tessellation Control shader (TCS) and Tessellation evaluation shader (TES).

In the TCS shader, Tessellation Levels can be set for each edge of the incoming patch. This parameter determines the degree of subdivision of each edge. In this paper, quadrilateral is used as the basic unit of Tessellation. The subdivision factor requires 4 edge (Outer) segmentation factors and two internal (Inner) segmentation factors [10], which correspond to the four sides of the quadrilateral and the horizontal and vertical subdivision. Figure 4 shows the subdivision results of quadrilateral elements under different subdivision factors.

Tessellation-based Grid Subdivision

To achieve adaptive Tessellation of the sea surface grid, the subdivision factors of each edge need to be calculated separately according to different conditions. In this paper, the subdivision factor of the corresponding edge is determined based on the distance between each vertex and the camera.

In addition to calculating the subdivision factor of each side of the Patch, it is also necessary to calculate the subdivision factor in the horizontal and vertical directions of the Patch. Here, in order to achieve a smooth transition inside the patch, the internal horizontal and vertical subdivision
factors are averaged by the subdivision factors of the upper and lower and left and right boundaries, respectively.

Currently OpenGL supports three subdivision modes, equal_spacing, fractional_even_spacing and fractional_odd_spacing. Equal_spacing means that each side of the patch is subdivided into equally divided segments, and the other two modes will achieve a smoother line segment length based on the subdivision factor of the floating point number. For the continuity of the grid after subdivision, the fractional_even_spacing mode is used here, and the subdivision effect diagram is shown in Figure 5.

![Figure 5. Type of partitioning of fractional_even_spacing.](image)

![Figure 6. Example of tessellation.](image)

**Experimental and Analysis**

The experimental PC's processor is Intel i5-3470, 8GB RAM, the graphics processor is Nvidia Quadro P2000, and OpenGL is used to build the experimental program in Visual Studio 2017 environment. The resolution of the display window is 1920 * 1080.

This experiment program first calculates the coordinates of 4 boundary points in the plane and horizontal plane in the CPU and transmits them to the GPU. The base grid is computed by the GPU based on the boundary point, and then the basic grid is subdivided in the subdivision shader in the GPU rendering pipeline to obtain the final grid data. Since a large amount of vertex calculation work is performed by a dedicated processing unit in the GPU, CPU resource consumption is reduced.

Figure 6 is an example of optimizing a grid using Tessellation. In this example, considering the direction of the cone and the distance between the grid and the observer, the program decides which areas of the surface are subdivided. Therefore, it can be seen that the degree of subdivision is not uniform in the grid model presented in the figure, and the grid area close to the observer is finer than the grid area far from the observer This allows for a more detailed picture in the vicinity of the observer, without wasting too much computing resources in the distance.

The base LOD grid size created by the program is 128*128, and then 3 levels are subdivided to get the final sea surface grid. Table 1 is a comparison of the performance of the proposed algorithm with the standard grid and concentric circular grid method [11]. It can be seen that the performance of this method is higher. In the case of drawing a similar number of triangles, the frame rate is higher and the CPU usage is lower.

This paper uses the Perlin noise[12] algorithm to simulate the ocean surface. The Figure 7 shows the sea surface effect simulated by the method. The ocean grid is only confined to the visible area and dynamically expands to the horizon as the camera pans or rotates. At the same time, using the adaptive continuous LOD strategy and Tessellation, the program realizes an automatic refinement grid from a distance to a viewpoint, and obtains a natural surface with both details and large scenes.

| Solution               | Number of triangles | FPS  | CPU  |
|------------------------|---------------------|------|------|
| standard grid          | 130050              | 53   | 43%  |
| Concentric-circular grid | 131232             | 73   | 35%  |
| Our solution           | 135424              | 85   | 24%  |
Conclusion

In the sea surface drawing, the GPU-supported hardware Tessellation technology can further refine the grid based on the CPU output rough grid, and realize the more natural and realistic natural scene simulation with limited computing resources. In this paper, an adaptive dynamic grid algorithm and a GPU programmable subdivision surface pipeline are used to propose an adaptive sea surface grid rendering algorithm. The algorithm can make full use of the GPU's hardware Tessellation performance, create a grid that can be efficiently drawn on the sea surface, and use Perlin noise to draw a more natural and realistic sea surface. Finally, the experiment shows that the method of this paper draws a good sea surface with good visual effect while reducing the consumption of computer resources.

Acknowledgement

This work was supported by the National College Students Innovation and Entrepreneurship Training Project (201710488003) and the College Students Science and Technology Innovation Foundation of Wuhan University of Science and technology (17ZRB112).

References

[1] Wang G, Ji Z Z, Zhang Z X, Real-time rendering of large scale dynamic ocean waves, J. Harbin Gongye Daxue Xuebao (Journal of Harbin Institute of Technology). 44(2012) 59-63.
[2] Zhao X, Pei B N, Fast ocean wave simulation method, J. Journal of System Simulation. 24 (2012) 132-131.
[3] Ren H X, Yin Y Y, Jin Y C, Realistic rendering of large-scale ocean wave scene, J. Journal of Computer aided Design & Computer Graphics. 20(2008) 1617-1622.
[4] Li B, Wang C, Li Z, et al. A practical method for real-time ocean simulation[C]/Computer Science & Education, 2009. ICCSE'09. 4th International Conference on. IEEE, 2009: 742-747.
[5] Belyaev V, Real-time simulation of water surface, J. Applied Math. Department, St. Petersburg State Polytechnical University, St. Petersburg, Russia, (2003) 131-138.
[6] Wang S L, Kang, F J, Xu J H, Real-time Rendering of Large-scale Coastal Water Based on Adaptive Optimization, J. Journal of System Simulation (S1004-731X), 27(2015) 2502-2507.
[7] Varela J M, Soares C G, Ring discretization of the wave spectrum for sea surface simulation, J. IEEE computer graphics and applications, 34(2014) 58-71.
[8] Su T, Cao Z, Lv Z, et al, Multi-dimensional visualization of large-scale marine hydrological environmental data, J. Advances in Engineering Software, 95(2016) 7-15.
[9] Weerasinghe M, Sandaruwan D, Keppitiyagama C, et al. A novel approach to simulate wind-driven ocean waves in the deep ocean[C]//Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on. IEEE, 2013: 28-37.

[10] Dong L, Zhang B, Zhao X, A Seamless Terrain Rendering Algorithm Based on GPU Tessellation, J. Geomatics & Information Science of Wuhan University, 42(2017) 402-407.

[11] Kryachko Y. Using vertex texture displacement for realistic water rendering, J. GPU gems, 2(2005) 283-294.

[12] Perlin K. An image synthesizer, J. ACM Siggraph Computer Graphics, 19(1985) 287-296.