An Efficient Algorithm for Rendering Large Bodies of Water

Ho-Min Lee, Christian Anthony L. Go, Won-Hyung Lee

Chung-Ang University,
Department of Image Engineering,
Graduate School of Advanced Imaging Science and Multimedia and Film,
221 Hukseok-Dong, Dongjak-Gu, Seoul, Korea
grancia@gmail.com, chipgo@gmail.com, whlee@cau.ac.kr

Abstract. Water rendering is one of the most computationally demanding task in computer graphics. Because of its computational complexity, real-time water rendering requires high-end hardwares. In this paper, we present a new algorithm for real-time rendering of large bodies of water such as open seas and oceans which results in an improved efficiency. Using interactive frustum, water surface can be fluidly calculated as a function of height given and interaction with another player in games. This results in an efficient yet realistic method for rendering large bodies of water without requiring as much computational power.

1 Introduction

Realistic water simulation technology has matured with the advent of advanced graphic hardwares. There are many limitations in using the CPU alone. With the advent of GPU's however, developers are now afforded added flexibility to render water in real time. In rendering realistic images of large bodies of water in games, there are four components that need to be addressed: atmospheric conditions, wave generation, light transport, and water surface size[1].
In games, there are many objects like players, NPC(Non Player Character), obstacles, buildings and so on, each of which interacts with one another. Generally, we need high computing power for handling the interaction between objects, and in this light, we should consider efficient computing powers. Real-time rendering also demands high computing power, making it more challenging to implement. In this paper, we describe an efficient approach for realistic large-scale water rendering for games.

2 Previous Work

2.1 Perlin Noise

The two most common algorithms employed for water simulation are the Navier-Stokes equation and Perlin Noise. The Navier-Stokes Equation is the cornerstone in the field of fluid mechanics and it can show realistic and complex phenomenon of fluid areas. However it is not appropriate because of high computational requirements. In this paper, we use Perlin Noise function for large water surfaces because it requires low computing power. Two functions are necessary to create the Perlin Noise, a noise function, and an interpolation function[3]. The Perlin Noise utilizes a random number generator and represents volume by interpolating random numbers.

2.2 Light Transport

To generate realistic simulation of natural water, one must consider in detail the interaction of light with the water body[4]. When light strikes the surface of a water, reflection and refraction phenomena occur on and under the surface. The most common method for the phenomena uses the Snell's law. Equation 1 demonstrates Snell's law.

$$sin\theta_1 \cdot n_1 = sin\theta_2 \cdot n_2 \tag{1}$$

where θ_1 is an incidence angle on water surface, θ_2 is a refraction angle under water surface, n_1 means air and n_2 means water. In this paper, we use shader programming to get efficient computing power by dividing the load between the CPU and GPU for light transport.

3 Water Surface Rendering

3.1 Represents Water Surface with Interactive Frustums

Many developers have tried to apply real-time water rendering to games. However, we still have a problem applying it to games because real-time water rendering requires high computing power. In this paper, we propose an efficient algorithm for rendering large bodies of water. This algorithm is based on the projected grid algorithm by Johanson[5]. He proposed a water simulation algorithm which is sufficient for water surface from a low viewpoint, however, the water surface cannot render smoothly when the viewpoint becomes higher. We improved on this algorithm specifically for ocean rendering in games. Fig. 1 shows how we define water surface sizes and interact with other characters. As you can see Fig. 1, when one player's frustum cross with another, height values of the surface frustums for water rendering are balanced automatically. As a result, players get the impression of being on the same water surface.

3.2 Water Rendering Range

We utilize two interactive frustums. One is for viewpoint, and another one is for definition of water surface sizes. In Fig. 1, as a height value of viewers, position of the frustum for water surface is defined. Equation 2 describes how the water

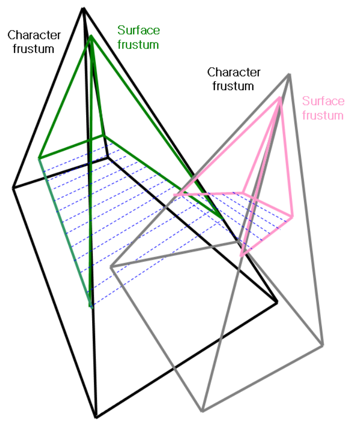

Fig. 1. Water rendering with interactive frustums

surface is defined as a function of height.

$$W_{size} = V_{HF} \cdot sin\theta \cdot W_{range} \qquad (\theta < 90) \tag{2}$$

where W_{size} is the water surface size, V_{HF} is the orthogonal height value from the water surface, W_{range} is the maximum range toward horizon and θ represents the angle between the surface and the user viewpoint. Thus, as a height value of characters, the water surface size is redefined within W_{range}. The traditional projected grid algorithm uses a grid for height value, and conversely, our improved algorithm calculates the grid size as an average of grid coordinates, resulting in improved speed.

4 Experimental Results

We experimented using Geforce 6600 GPU, Pentium 4 processor running at 3.0 GHz. Fig. 2 shows the experimental results in terms of different height values: 80, 1000 and 2000 pixels from top-down. In the Fig. 2, the projected grid algorithm shows that cuts the water surface and displays a part of the earth's surface at approximately 2000 pixel-height. However, although when the player's height value is increased, our proposed algorithm still renders realistic water surface efficiently. Moreover, we added objects in space, and it showed increased computing efficiency by about 30%.

5 Conclusion

We have presented a method for an efficient rendering of large bodies of water. The method uses an interactive and flexible frustums, Fig. 1 shows how the water

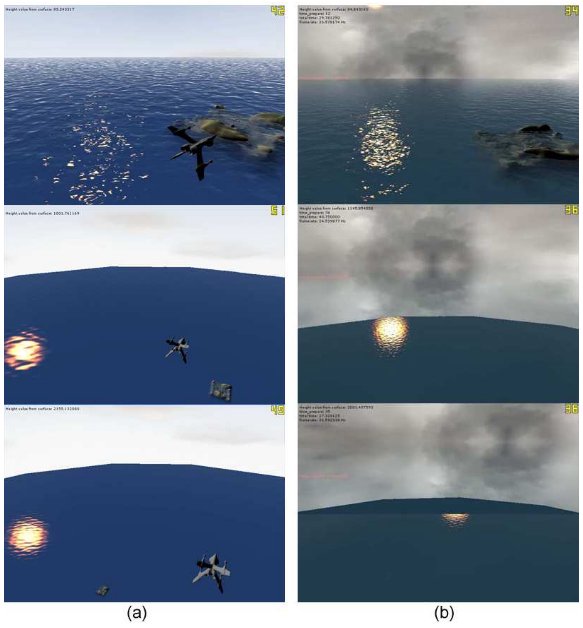

Fig. 2. (a) The proposed algorithm (b) Projected grid algorithm.

surface is defined in terms of height value and interaction with another player. Experimental results have shown our algorithm to render realistic large bodies of water while being flexible when interacting with other players. Moreover, we achieved a 30% increased efficiency in terms of computational costs.

Acknowledgment

This research was supported by the Korea Culture and Contents Agency, and the Research Grant of Chung-Ang University, Seoul, Korea.

References

1. Simon Premoze, Michael Ashikhmin.: Rendering Natural Waters. IEEE Computer Society. (2000) 23
2. Jim X. Chen , Niels da Vitoria Lobo.: Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. Graphical Models and Image Processing. **57** (1995) 107–116
3. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley.: Texturing & Modeling - A Procedural Approach Second Edition. AP Professional. (1998)
4. Alain Fournier, William T. Reeves.: A Simple Model of Ocean Waves. ACM Press. **20** (1986) 75–84
5. Claes Johanson.: Real-time water rendering. Lund University. (2004)