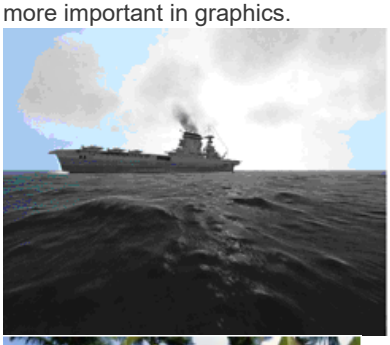
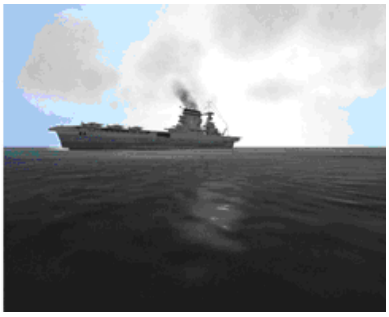
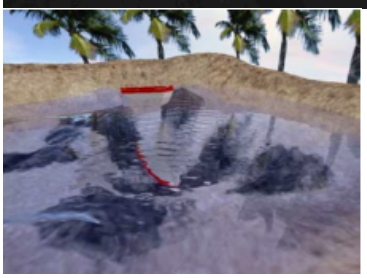
Computer Graphics
NTUST CSIE Laboratory

Project 4: Water Surface and Rendering (GPU Shader)

Introduction

Water is common seen in daily life. However, the interaction and rendering of water involve complex physical computation. In this work, we would like you to explore a few possibility to simulate the water surface and render the surface with shader programs and CUDA programs which become more and more important in graphics.

The Basic Task

With the advance of GPU technology, GPU becomes more and more popular in computer graphics specially in game industry. It is also noticed by general computation community. Water surfaces are common in computer graphics, especially in games for creating fantastic effects. They are a critical element that can significantly improve the level of realism in a scene. But depicting them realistically is a hard problem, because of the high visual complexity present in the motion of water surfaces, as well as in the way light interacts with water. This project would give you the chance to explore techniques developed for rendering realistic depictions of the water surface. With these two things in mind, this project would like you to have the experience by implementing a shader program and a CUDA program to generate a water surface and then rendering it with reflection and refraction effects. In addition, ray-tracing is very important global illumination algorithm to generate realistic images in graphics community. Therefore, the project also ask you to implement a ray tracer for rendering the water and other scene objects.

Please print Project4-Grading.doc, let TA to check your score.

You can download Project4_Solution.zip to execute.

The Tasks

Fundamentally, to generate the perception of the water in an interactive application consists of two tasks:

1. Your program has to generate a water surface according to some physical rules which are formed for some specific phenomenon in your mind. This generated surface represents the boundary of water for a renderer to generate proper images for illustrating the water surface.
2. Your program must render the water surface to simulate the effects of refraction and reflection. Furthermore, the caustics caused by water movement on the sea floor can be added to add more reality.

In addition, realistically rendering the water surface is also important for other graphics applications, a basic ray-tracing algorithm is required to simulate the global illumination effect in the scene. The ray tracer will generate an image according to rays through the center of all pixels to interact with the scene object and then compute the lighting effects and shadow effects.

Functions		score
Water surface	Sine waves	30
	Interactive wave-function-based height maps	30
Shader-based GPU Rendering	Skymapping reflection	30
	Multi-passed scenic refraction and reflection	30
	Caustics effect	30

Water surface generation with GPU

Basically, the water surface can be simply decomposed into two geometric components: high-level surface structure and low-level surface details.

1. The surface structure represent the large scales of the wave movement and can be represented with a set of 2D grid with the y direction represent the height of the grid point. A vertex shader can be used to simulate the movement at the vertices by changing the normal and the y position of the vertex.
2. The surface detail represent the small scale perturbation in the local area and generally can be represented as a normal map. A pixel shader is created to generate a normal map for those perturbation.

Generally, there are several ways to implement the wave on the surface of water:

1. Sine waves (20) (Ref. 1): The sum of the sine waves are chosen to represent the complex water surface movement. The equation can be expressed as

$$W_i\left(x,y,t\right) = A_i \times \sin\left(\mathbf{D}_i \cdot \left(x,y\right) \times w_i + t \times \varphi_i\right).$$

$$H\left(x,y,t\right) = \sum\left(A_i \times \sin\left(\mathbf{D}_i \cdot \left(x,y\right) \times w_i + t \times \varphi_i\right)\right),$$

   - Wavelength (L): the crest-to-crest distance between waves in world space. Wavelength L relates to frequency w as w = 2w/L.
   - Amplitude (A): the height from the water plane to the wave crest.
   - Speed (S): the distance the crest moves forward per second. It is convenient to express speed as phase-constant, phi , where phi = S x 2w/L.
   - Direction (D): the horizontal vector perpendicular to the wave front along which the crest travels. Please refer to Ref. 1 for more direction details.

1. Height maps (20) (Ref. 3): similar to sine wave method, height map method decomposes the wave on the water surfaces into a set of different level of detail represented as a heightmap (the shape of single component and generated by artists) The following shows an example of the height map.

The combination of the heightmap can be expressed as the following equation.

$$H(x, y, t) = \sum_{i=0}^{N} h\left(A_i^x x + B_i^x,\ A_i^y y + B_i^y,\ A_i^t t + B_i^t\right).$$

Please refer to Ref. 3 for more heightmap details.

1. Wave equation(Ref. 4): Generally, the movement of the wave can be expressed as a wave equation as listed in the following:

$$\frac{\partial^2 y}{\partial t^2} = c^2\left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2}\right)$$

Then by modeling the water surface as cubic Bspline surfaces, the right hand side of the equation can be expressed by the following:

$$c^2\left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2}\right) = c^2([y_{-1,0} + y_{1,0} - 2 \cdot y_{0,0}] + [y_{0,-1} + y_{0,1} - 2])$$

$$c^2\left(\frac{\partial^2 y}{\partial x^2} + \frac{\partial^2 y}{\partial z^2}\right) = c^2(y_{-1,0} + y_{1,0} + y_{0,-1} + y_{0,1} - 4 \cdot y_{0,0})$$

And then the integration can be computed with one of the following integration methods

$$p(t_2) = 2 \cdot p(t_1) - p(t_o) + \frac{1}{2} \cdot a(t_1) \cdot h^2$$

$$p(t_2) = 2 \cdot p(t_1) - p(t_o) + a(t_1) \cdot h^2$$

$$p(t_2) = (1 + \alpha) \cdot p(t_1) + \alpha \cdot p(t_o) + \frac{1}{2} \cdot a(t_1) \cdot h^2$$

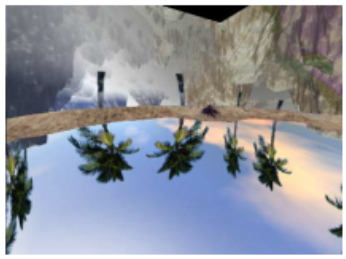
Then the equation is solved at each vertex of the 2D mesh to generate the water surface. Please refer to Ref. 4 for details.

Water surface rendering with GPU

Two important effects for the water surface rendering: reflection and refraction. We assume that we are looking outside the water. The reflection and refraction can be generated with the following different methods

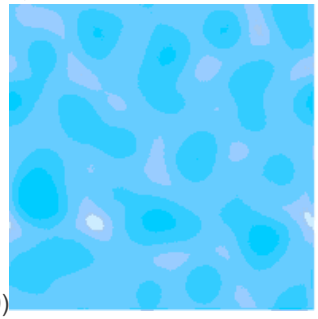
1. Combination of refraction and reflection with an environment map for the sky and a set of texture map for the box. (Ref. 1) (20)
2. Using the refraction map and reflection map to simulate the possible viewing condition from above the water to simulate the refraction and reflection effects. (Ref. 4) (25). The following shows an example of refraction and reflection map

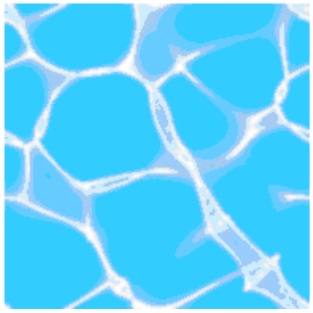
Refraction Map



Reflection Map

In addition to the refraction and reflection, the caustics on the floor due to concentration of refraction and reflection is also visually important. Therefore, simulate the caustics effect on the floor is also important for the rendering. (20)





What to hand in?

As usual, you must hand in everything needed to build and run your program, including all texture files and other resources.

In your readme, please make sure to have the following (you can break it into seperate files if you prefer):

1. Instructions on how to use your program (in case we want to use it when you're not around)
2. Descriptions of what your program does to meet all of the minimum requirements.

You should make a subdirectory of the project directory called "Gallery." In this directory, please put a few JPG pictures of the best scenes in your town. Please name the pictures login-X.jpg (where X is a number). Put a text file in the directory with captions for the pictures. (note: to make pictures, use the screen print and then use some program to convert them to JPG). ?

Reference

1. GPU Gems Chapter 1: Effective Water Simulation from Physical Models
2. GPU Gems Chapter 2: Rendering Water Caustics
3. GPU Gems II Chapter 18: Using Vertex Texture Displacement for Realistic Water Rendering
4. Vertex Texture Fetch Water:
    1. Source Code from NVidia
    2. User guide from NVidia