Numerical Analysis Project 4

Margaret Dorsey

December 9, 2016

Introduction

The current gold standard of non-interactive large body waves is set by a paper of Tessendorf, which is available among the submission files as *simulating-ocean-water_tessendorf.pdf*. The goal of this project is to implement his method, based on the Fast-Fourier Transform, to create a convincing representation of ocean water by deforming a planar surface.

Derivation

Note: This derivation is essentially a summary of the work of Jerry Tessendorf and Keith Lantz, and should not be interpreted as my own work.

Let $\vec{x} = (x, y, z)$ be the position vector of a given vertex of our plane Q, with Q having width N, depth M, and a height of 0, centered about the origin $(0, 0, 0)$. We then define the equation for wave height h as

$$h(\vec{x}, t) = \sum_{\vec{k}} \tilde{h}(\vec{k}, t) e^{i\vec{k}\cdot\vec{x}}$$

where t is time, and

$$\vec{k} = \left(\frac{2\pi n}{\vec{x}_x}, \frac{2\pi m}{\vec{x}_z}\right)$$

$$\frac{-N}{2} \leq n < \frac{N}{2}, \frac{-M}{2} \leq m < \frac{M}{2}$$

To facilitate the use of indices rather than coordinates, we introduce n' and m', where

$$0 \leq n' < N, 0 \leq m' < M$$

$$n = n' - \frac{N}{2}, m = m' - \frac{M}{2}$$

This gives us a new, equivalent function for wave height,

$$h'(\vec{x}_x, \vec{x}_z, t) = \sum_{m'=0}^{M-1} \sum_{n'=0}^{N-1} \tilde{h}'(n', m', t) e^{i\vec{k}'\cdot\vec{x}}$$

We then define $\tilde{h}$ as follows:

$$\tilde{h}(\vec{k},t) = \tilde{h_0}(\vec{k}e^{i\omega(\vec{k}t)} + \tilde{h_0}^*(-\vec{k})e^{-i\omega(\vec{k})t}$$

where $\omega(\vec{k}) = \sqrt{g\vec{k}}$ is the dispersion due to gravity, and $\tilde{h_0}$, the complex conjugate, is

$$\tilde{h_0}(\vec{k}) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h(\vec{k})}$$

with ξ_r, ξ_i are independent random numbers, and P_h is the Phillips spectrum modelling wind given by

$$P_h(\vec{k}) = A\frac{e^{-1/(\vec{k}L)^2}}{\vec{k}^4}|\vec{k}\cdot\vec{w}|^2$$

where $\vec{w}$ is the direction of the wind, and A is the maximum height of a wave with wind at a speed of L.

In addition to our height function h, we create a displacement function d to create "choppiness" in our waves, and a normal vector displacement n.

$$d(\vec{x},t) = \sum_{\vec{k}} -i\frac{\vec{k}}{k}\tilde{h}(\vec{k},t)e^{i\vec{k}\cdot\vec{x}}$$

$$n(\vec{x},t) = (0,1,0) - (\epsilon_x(\vec{x},t),0,\epsilon_z(\vec{x},t)) = (-\epsilon_x(\vec{x},t),1,-\epsilon_z(\vec{x},t))$$

$$\epsilon(\vec{x},t) = \nabla h$$

Implementation

The implementation of the fast fourier transform of the system of two one-dimensional transforms that arise from the above two-dimensional discrete transform can be found within the ocean class, in *ocean.h* and *ocean.c*. This implementation is based heavily on the implementation by Keith Lantz and the implementation of FFT found in the textbook *Numerical Recipes in C: The Art of Scientific Computing*

Essentially, the displacement of each vertex is calculated, and passed to the rendering pipeline as a transformation matrix to be applied to each vertex as it is processed for screen rasterization. This is not the most optimal data flow, but it is one that is simple enough for the scope of this project.

Outputs and Executables

The following is a gallery of screenshots taken from the executable of the submitted program.

The program can be built using the submitted files and run to see the real time animation allowed by the FFT implementation, however it requires an OpenGL version

not guaranteed by integrated graphics cards, so compatibility issues in running may arise. The program uses the glm math library and the SOIL image library. GLM is header-only, and thus very portable, however to compile with SOIL you must download and build SOIL from source on your machine.

Two executable files, one for Unix-based systems and one for Windows-based systems, have been included. I do not promise that either of them will run on any given machine except the ones that I developed on.

A video file of the program at runtime has been included.

Results and Notes

The FFT implementation did indeed produce a realistic graphical simulation of ocean waves, as desired. Future improvements would involve offloading more of the computational work to the GPU, more realistic calculation of reflection/refraction of light, and integration into a more complex application, such as a game or simulation.