CS175 - Final Project

Victoria Kauffman

May 2023

1 General Premise

For this assignment, I wanted to figure out how to make realistic ocean movements. In other words, I sought to create a realistic ocean scene with ripples and wave movements.

Through this process, I looked at multiple kinds of waves as I sought to find the one that suited my needs best. I will discuss my experience with Gerstner waves and with FFTs / Oceanographic Spectra. My final project that I submitted was the one that uses FFTs.

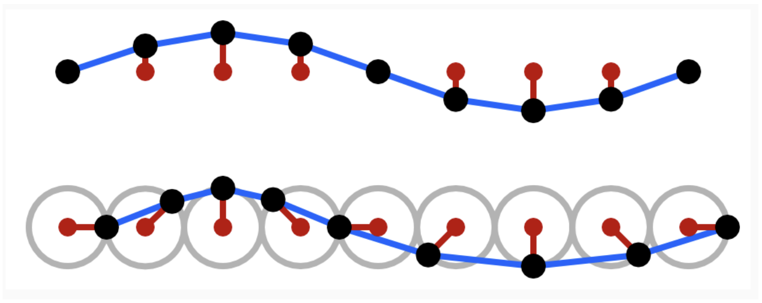
2 Gerstner Waves

** The information in this section primarily comes from Catlike Coding **

Gerstner waves rely on the periodic nature of larger waves. They are very similar to sin waves, but they have sharper peaks and deeper troughs to be more accurate to the imperfect shape of moving water.

The image below shows the difference in the movement of a surface point across sine vs Gerstner waves:

Figure 1:



(Catlike Coding)

Computing Gerstner waves requires a fair amount of math. At its basis, the math primarily expands upon traditional sine waves. It relies on a normalized direction vector, D.

We can have a wave number $k = \frac{2\pi}{\lambda}$, where λ is the wavelength.

We can then consider some variable A to represent the wave amplitude. It is equal to the "steepness" of the wave divided by k.

We want to consider the offsets of each point of our wave as a function of time. An equation that models this value is

$$P = \begin{bmatrix} x + D_x A \cos f \\ A \sin f \\ z + D_z A \cos f \end{bmatrix}$$

where f is a function of time (namely, $f = k(D \cdot \begin{bmatrix} x \\ z \end{bmatrix} - ct)$, where $c = \sqrt{\frac{9.8}{k}}$ and t reflects the time that has passed. Incorporating considerations of gravity through c, time, and the properties of the wave itself, this equation provides a relatively decent look at an individual wave within the ocean.

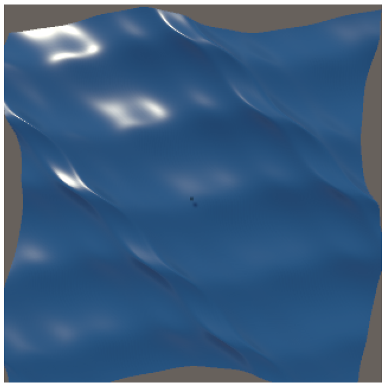
Finding the normal of Gerstner waves follows a similar process. To find it, we can take the normalized cross product of the binormal and tangent. Finding each of those terms relies on taking partial derivatives. If we have s signify the "steepness" of our wave, we find that the binormal is equal to

$$\begin{bmatrix} -D_x D_z s \sin f \\ D_z s \cos f \\ 1 - D_z^2 s \sin f \end{bmatrix}$$

and the tangent is equal to

$$\begin{bmatrix} 1 - D_x^2 s \sin f \\ D_x s \cos f \\ -D_x D_z s \sin f \end{bmatrix}$$

Using these equations, I thus created a shader to design waves in Unity. I added the ability to overlay multiple waves by summing the offsets they caused in position by each other. Playing around with the wavelengths and other variables, I created the following rippling image:



While creating this image was a satisfying process, I wanted to investigate more realistic-looking waves.

3 Realistic Water

** The basis of my understanding of this area comes from a Jump Trajectory YouTube video called "Ocean waves simulation with Fast Fourier transform" and a published paper by Christopher J. Horvath titled "Empirical Directional Wave Spectra for Computer Graphics." **

As I researched more realistic water, I learned about FFT and oceanographic spectra. In essence, this strategy relies on the idea that more realistic water is made possible by repeatedly creating countless ripples and waves within a mesh.

3.1 Oceanographic Spectra

** I drew on Horvath's "Empirical Directional Wave Spectra for Computer Graphics" for this section "

Oceanographic Spectra, in essence, describes a process to choose logical positions from which to initialize the waves used in FFTs. There are a variety of different spectra that one can use, all of which can produce realistic-looking waves able to react to a variety of different conditions (including wind, swells, etc). Some spectra include the Pierson-Moskowitz Spectrum, the Tessendorf Spectrum, the JONSWAP Spectrum, and the Texel MARSEN ARSLOE (TMA) Spectrum.

I chose to first use TMA Spectrum. THE TMA Spectrum incorporates the JONSWAP Spectrum and an additional factor to make the Spectrum more dynamic in water of various depths.

3.1.1 The JONSWAP Spectrum

The JONSWAP Spectrum's overall equation follows:

$$\frac{\alpha g^2}{\omega^5} \exp(-\frac{5}{4}(\frac{\omega_p}{\omega})^4 \gamma^r$$

where γ^r represents a peak-enhancement factor and we have, in general:

$$r = \exp - \frac{(\omega - \omega_p)^2}{2\sigma^2 \omega_p^2}$$

$$\alpha = 0.076(\frac{U^2}{Fg})^{0.22}$$

$$\omega_p = 22(\frac{g^2}{UF})$$

$$\gamma = 3.3$$

$$\sigma = \begin{cases} 0.07 & \omega \leq \omega_p \\ 0.09 & \omega > \omega_p \end{cases}$$

This Spectrum incorporates several factors in an attempt to match empirical data collected in the Joint North Sea Wave Observation Project (JONSWAP). In the equations above, U represents wind speed, g represents gravity, F represents fetch, and ω represents the dispersion relations between the waves. In general, U, g, and F are some constants dependent on the setting, while ω can be found as follows:

Let us have some h represent depth, λ represent wavelength, and $k = \frac{2\pi}{\lambda}$. Then, according a Wikipedia article, we have:

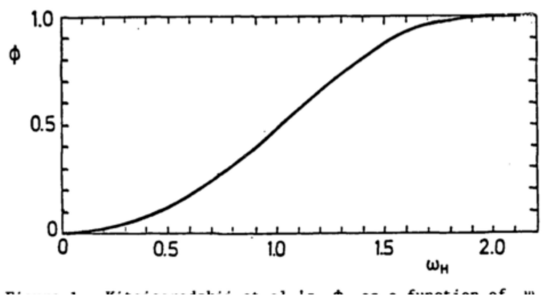
$$\omega = \begin{cases} \sqrt{gk} & h > \frac{1}{2}\lambda \\ k\sqrt{gh} & h < 0.05\lambda \\ \sqrt{gk \tanh(kh)} & \text{otherwise} \end{cases}$$

We can thus use these values to find our JONSWAP factor.

3.1.2 TMA

To fully compute TMA, our final value will be $S_{JONSWAP}\Phi$, where $S_{JONSWAP}$ is the value found via the JONSWAP Spectrum. We find Φ as follows:

We can find the following graph by plotting Φ against ω_h, where $\omega_h = \omega\sqrt{h/g}$:



(Horvath)

We can reduce this value with less than 4% error using the formula below:

$$\Phi = \begin{cases} \frac{1}{2}\omega_h^2 & \omega_h \leq 1 \\ 1 - \frac{1}{2}(2 - \omega_h)^2 & \omega_h > 1 \end{cases}$$

We thus find the product of $S_{JONSWAP}$ and Φ to find our value.

3.1.3 Directional Spreading

We can then incorporate an understanding of how wave energy disperses by considering directional spreading. As above, there are several ways to calculate directional spreading, but I chose to use an interpolation of Positive Cosine Squared Directional spreading and Donelan-Banner, since I found those values yielded realistic-enough waves with relatively simple equations.

3.1.4 Positive Cosine Squared Directional spreading

Positive Cosine Squared Directional spreading has no reliance on wavelengths, meaning that all waves at a certain angle from the wind θ are elongated equally. Its value is found through the following equation:

$$\begin{cases} \frac{2}{\pi}\cos^2\theta & \frac{-\pi}{2} < \theta < \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

3.1.5 Donelan-Banner Directional Spreading

Donelan-Banner Directional spreading once again relies on empirical data. It first uses ω and ω_p (where $\omega_p = 22(\frac{UF}{g^2})^{0.33}$)to determine a β as follows:

$$
\begin{aligned}
\beta_s &= \begin{cases} 2.61(\omega/\omega_p)^{1.3} & \text{for } 0.56 < \omega/\omega_p < 0.95 \\ 2.28(\omega/\omega_p)^{-1.3} & \text{for } 0.95 \leq \omega/\omega_p < 1.6 \\ 10^{\epsilon} & \omega/\omega_p \geq 1.6 \end{cases} \\
\epsilon &= -0.4 + 0.8393\exp[-0.567\ln((\omega/\omega_p)^2)]
\end{aligned}
$$

This β is the used to find our Directional spreading factor, represented as

$$: \frac{\beta_s}{2\tanh(\beta_s\pi)}\operatorname{sech}(\beta_s\theta)^2$$

All of these processes come together to allow us to compute starting values for our waves in the form of

$$= \sqrt{2S(k_x, k_y)\Delta k_x \Delta k_y}$$

[illegible]

where S is the product / sum of all spectrums.

3.2 FFT

Now that we have calculated these values, we can use them to help us approach FFT. FFT, or Fast Fourier Transform, is a way of performing Discrete Fourier Transform faster. In general, Fourier Transforms take a time-based pattern (for instance, the movement of an ocean wave) and output the predicted cycles of this pattern. Discrete Fourier Transform can compute the location of every particle in every wave, but this process would take much too long (up to $O(MN)$, where M and N refer to the number of waves and the number of particles per wave).

Instead, FFT can be used to compute the data in $O(N \log N)$ time. The process basically iteratively finds predicted locations for the data using both the direct and inverse FFTs.

4 Math Resources

To understand and figure out how to implement Gerstner Waves, I used a 2018 Catlike Coding tutorial located at the following link:
https://catlikecoding.com/unity/tutorials/flow/waves/.

This tutorial provided a comprehensive overview of not only the equations relating to Gerstner Waves, but also the process of their implementation in Unity.

My discussion of Oceanographic Spectra relied heaviliy on Horvath's "Empirical Directional Wave Spectra for Computer Graphics," available at this link:

https://dl-acm-org.ezp-prod1.hul.harvard.edu/doi/pdf/10.1145/2791261.2791267

I supplemented this reading with Jump Trajectory's video to gain a better understanding of the technique.

I also used a Wikipedia article on Dispersion in order to compute ω. It is located at the following link:

https://en.wikipedia.org/wiki/Dispersion_(water_waves)

To implement and understand FFT, I primarily used Jump Trajectory's "Ocean waves simulation with Fast Fourier transform" YouTube video. It is located at the following link:

https://www.youtube.com/watch?v=kGEqaX4Y4bQ

To bolster my understanding of FFTs, I also reviewed the following article:

https://towardsdatascience.com/fast-fourier-transform-937926e591cb

5 Coding Resources

Due to my desire to focus on the mathematics behind these concepts rather than on figuring out Unity, I used the YouTube video listed above, as well as the code base linked to it, to handle my scripting / building of meshes. Once again, the link to the YouTube video is

https://www.youtube.com/watch?v=kGEqaX4Y4bQ

and the GitHub that held the code I used for the less exciting portions (including sending variables to my various shaders, a SimpleCameraMover, and the material shader) is located at the link below:

https://github.com/gasgiant/FFT-Ocean

The creator of this code base is GasGiant.

6 Hot Keys

W, A, S, D will move the camera forward / left / backward / right, while Q and E will raise and lower it.

O slows down the local wind speed
P speeds up the local wind speed
K slows down swell wind speed
L speed up the swell wind speed
U decreases the local wind direction
I increases the local wind direction
H decreases the swell wind direction
J increases the swell wind direction

7 Running the Program

I included in my project a “Build” file containing an large number of options for building the program on a variety of OS, including Mac, Windows 32 bit, Windows 64 bit, and Linux.