Ocean Generation in Modern Games

Ong Wei Yong
20022191

University of the West of England

April 14, 2023

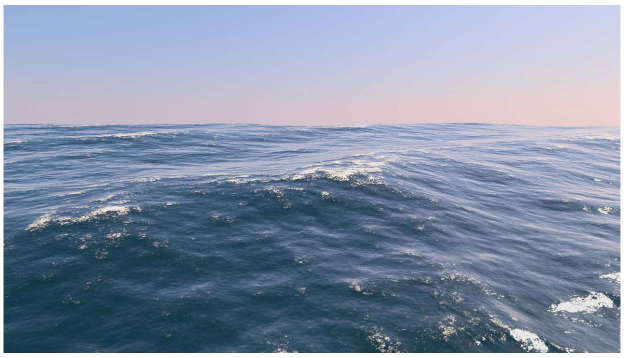

Figure 1: *Fast Fourier Ocean*

This report elaborates on an implementation to best simulate water. In order to most accurately represent water, an ocean in Unity Engine was created using Fast Fourier Transform algorithms. By using FFT algorithms to simulate the behavior of waves, water surfaces can be generated to respond realistically to changes in wind and other environmental factors.

1 Introduction

With the increasing demand for realism in video games, many game studios and developers have been researching and implementing different methods to achieve realistic oceans in modern video games. One such method utilised is using Fast Fourier Transform algorithms. An example of an implementation can be seen in Figure 1.

1.1 Research Background

Several methods of simulating water and oceans were researched. The two most prominent methods to simulate oceans were by using Gerstner Waves Method which was discovered by Gerstner, 1809, as well as using the Fast Fourier Transform Algorithm Ocean by Tessendorf et al., 2001. Gerstner Waves, also known as Trachoidal Waves, are an exact solution of the Euler equations for periodic surface gravity waves. They describe a progressive wave of permanent form on the surface of an incompressible fluid of infinite depth. Williams, 2019 describes the method as a modified Sine Wave. However, unlike a sine wave it has sharper peaks and flatter valleys (where as a sine wave has identical peaks and valleys). The Fast Fourier Transform algorithm is described by Heideman, Johnson, and Burrus, 1984 as an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis is a method that transforms a signal from its original domain (such as time or space) to a representation in the frequency domain, and vice versa. The Discrete Fourier Transform (DFT) involves breaking down a sequence of values into various components of different frequencies. In terms of computer graphics and ocean simulations, FFT algorithms can be used to generate a series of 2D height maps that represent the surface of the ocean at different points in time. These height maps are then combined to create a 3D mesh that accurately represents the shape and motion of the ocean surface as demonstrated by Tessendorf et al., 2001.

Ultimately, the Fast Fourier Transform Method was selected for this implementation for the following reasons:

- FFT Oceans look more realistic compared to Gerst-

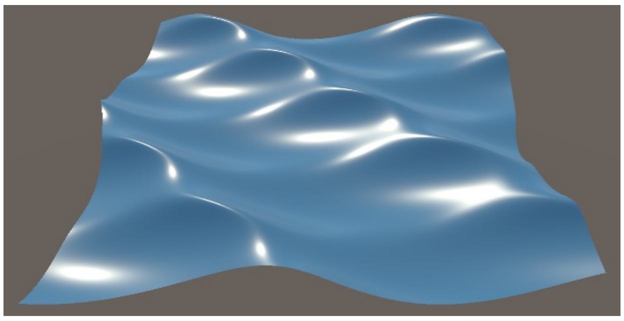

Figure 2: *Gerstner Waves*

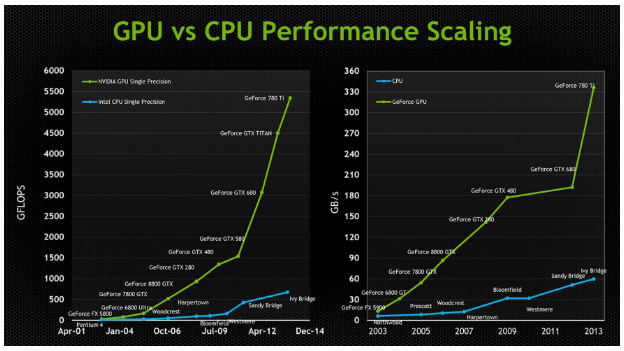

Figure 3: *Comparison of CPU and GPU performance measured in GFLOPS and GB/s over the past years.*

ner Waves. Gerstner Waves have larger and more stylized looking waves as shown in Figure 2 compared to the more realistic FFT Ocean.
- FFT Oceans look more randomized and less uniform compared to Gerstner Waves.
- More realistic wind as well as foam can be simulated using the FFT Method compared to the Gerstner Waves method.

Research for the implementation method was also done. As explained by Flügge, 2017, the FFT's in Crysis (Crytek, 2007) were generated on CPU with a generated thread. The use of high resolutions posed a problem in real-time water rendering because it required the computation of one or three FFTs per frame, depending on whether Choppy-displacement was applied. To address this issue, the computation was moved to the GPU, which offers a parallel architecture with many small cores that can handle multiple tasks simultaneously. This is in contrast to a CPU with a limited number of cores that processes tasks sequentially, making processing on the GPU faster than the CPU as shown in Figure 3. Since 2D-FFT is easily parallelizable, the GPU provides a solution to the performance problem. In modern Game Engines, GPGPU became essential for FFT generated water. Thus, the current implementation of this study utilizes compute shaders on the GPU for FFT algorithm processing as a result of the research conducted.

2 Related Work

Tessendorf et al., 2001 introduced an FFT-based ocean wave simulation which used a choppy effect to form the dramatic wave shape. Wang et al., 2011 have implemented a simulation program on a GPU running in real time.Wang et al., 2011 implemented a FFT ocean in OpenGL. Suriano, 2017 presented his implementation of his FFT Ocean in Rome. Nvidia, 2015 presented their implementation of an FFT Ocean titled Ocean Surface Simulation.

Listed are some games and movies that have utilised FFT Algorithms for their Ocean Simulation:

- War Thunder (Gaijin Entertainment, 2012)
- Sea of Thieves (Microsoft Studios, 2018)
- Assassins Creed 3 (Ubisoft, 2012)
- Assassins Creed Black Flag (Ubisoft, 2013)
- Crysis (Crytek, 2007)
- Moana (Walt Disney Animation Studios, 2016)
- Titanic (20th Century Fox, 1997)

3 Method

This study was implemented on the Unity Engine Version 2022.1.16.f1 with the Universal Render Pipeline. Compute Shaders were utilized in this implementation to calculate the FFT Algorithm.

3.1 Ocean Waves

In this implementation, Fast Fourier Transform algorithms were utilised to visually simulate the ocean. The simulation takes in a frequency domain spectrum representation and performs the inverse of the Discrete Fourier Transform (DFT) of the spectrum. The result is the time domain or heightmap at time $\boldsymbol{t}$.

Suriano, 2017 lists several reasons of utilising FFT algorithms instead of DFTs for ocean simulations.

- Computing the DFT on its own is expensive.
- DFT Complexity is $O\left(n^2\right)$.
- FFT algorithms instead have a complexity of $O\left(n\log_2(n)\right)$.

For this implementation, the Cooley-Tukey (Butterfly) FFT algorithm was used.

For realistic simulations, Tessendorf et al., 2001 mentions that Oceanographic literature tend to utilise statistical ocean models. Wang et al., 2011 adds that statistical models are also based on the ability to decompose the wave height field as a sum of sine and cosine waves. Inverse FFT algorithms can then composite many sine and cosine waves to form the wave surface.

$$h(\mathbf{x},t) = \sum_{\mathbf{k}} \tilde{h}(\mathbf{k},t)\exp(i\mathbf{k}\cdot\mathbf{x}). \tag{1}$$

Tessendorf et al., 2001 explains Equation 1 as follows: The height $h(\mathbf{x}, t)$ at a horizontal position $\mathbf{x} = (x, z)$ at time t is defined by the sum of sinusoids with complex amplitudes. The wave vector $\mathbf{k}$ is a two-dimensional horizontal vector, which points in the direction of travel of the wave. The components k_x and k_z of $\mathbf{k} = (k_x, k_z)$ are defined as:

$$k_x = (2\pi n/L)$$

and

$$k_z = (2\pi m/L)$$

where n and m have bounds:

$$-N/2 \leq n, m < N/2$$

$\mathbf{n}$ represents the horizontal domain resolution where as $\mathbf{m}$ represents the vertical domain resolution. Tessendorf et al., 2001 recommends resolution sizes in between 16 and 2048, in powers of two. Additionally, ranges between 128 and 512 are sufficient for simulations. In this implementation, both $\mathbf{n}$ and $\mathbf{m}$ are set as **256**.

Using an inverse FFT, a height field can be generated at each point $\mathbf{x} = \left(\frac{nL_x}{N}, \frac{mLz}{M}\right)$ for $\mathbf{n}$, $\mathbf{m}$.

Joining it all together will result in this equation: $h(\vec{x}, t) = h(x, z, t) =$

$$\sum_{m=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \tilde{h}(\frac{2\pi n}{Lx}, \frac{2\pi m}{Lz}, t) e^{i(\frac{2\pi}{Lx}x + \frac{2m}{Lz}z)} \qquad (2)$$

In order to get $\tilde{h}(\mathbf{k}, t)$, Tessendorf et al., 2001 provided the Phillips Spectrum as shown in the following equation:

$$P_h(\mathbf{k}) = A \frac{\exp\left(\frac{-1}{(kL)^2}\right)}{k^4} |\hat{\mathbf{k}} \cdot \hat{w}|^2 \qquad (3)$$

where

$$L = V^2/g$$

L represents the maximum waves height. V is the wind speed. $\hat{w}$ represents the wind direction and A is a constant. g represents the gravitational constant of the earth where $g = 9.8m/sec^2$. The term $\hat{\mathbf{k}} \cdot \hat{w}|^2$ removes waves that move vertically to the direction of the wind.

Tessendorf et al., 2001 also elaborates how the model has poor convergence properties at high values of wave number $|k|$. A fix to this issue is by suppressing very small waves where $l << L$. This is done by multiplying the Phillips Spectrum with the factor $exp(-k^2 l^2)$.

This would result in the following equation:

$$P_{h_{\text{modify}}}(\vec{k}) = P_h(\vec{k}) \times e^{-k^2(0.001L)^2} \qquad (4)$$

Fourier amplitudes of a wave height field can then be computed with the Phillips Spectrum using the following equation:

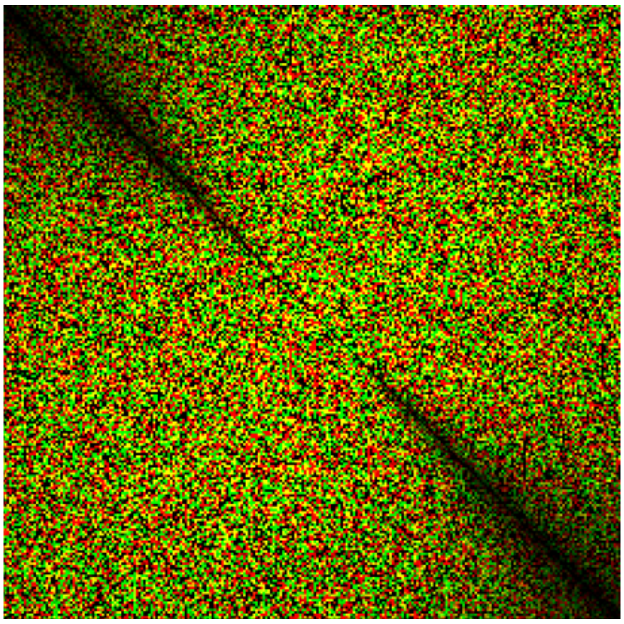

Figure 4: $\tilde{h}_0(\mathbf{k})$ *texture*

$$\tilde{h}_0(\mathbf{k}) = \frac{1}{\sqrt{2}} (\xi_r + i\xi_i) \sqrt{P_h(\mathbf{k})} \qquad (5)$$

ξ_r and ξ_i are independent numbers of a gaussian normal distribution with mean 0 and standard deviation 1.

The dispersion relation:

$$w^2(k) = gk \qquad (6)$$

is then integrated into the equation of Fourier amplitudes at Equation 5. The dispersion relation is described as the relationship between frequencies and magnitudes of the water wave vectors k_i where g is the gravitational constant.

The wave amplitudes at time t can then be calculated as shown in the following equation.

$$\tilde{h}(\mathbf{k}, t) = \tilde{h}_0(\mathbf{k}) \exp(iw(k)t) + \tilde{h}_0^*(-\mathbf{k}) \exp(-iw(k)t) \qquad (7)$$

Given that the Euler formula is: $e^{ix} = cos(x) + isin(x)$, we can derive the following equation:

$$e^{iw(k)t} = cos(w(k)t) + isin(w(k)t) \qquad (8)$$

3.2 Gaussian Noise

To attain ξ_r and ξ_i as shown in Equation 5, the Box Muller transform was used to get the Gaussian noise as shown in Figure 4 as normally distributed random numbers were required. Dialid, 2019 mentions that the transform takes two samples from the uniform distribution on the interval and maps them to two standard, normally distributed samples.

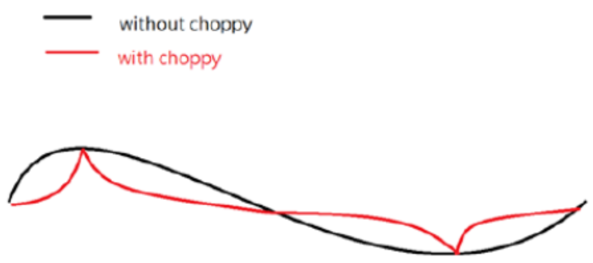

Figure 5: *Choppy effect*

The implemented method is as follows: U_1 and U_2 were generated as independent random variables distributed uniformly on [0,1].

R was then defined as:

$R = \sqrt{-2 \cdot InU_1}$ and $\theta = 2\pi U_1$.

Following that,

$Z_1 = R * \cos\theta$

$Z_2 = R * \sin\theta$

would then be independent random variables with standard normal distribution.

The pseudo code implementation is as follows:

```
    u1 = random(n)
    u2 = random(n)

    r = sqrt(-2*log(u1))
    x = cos(2*pi*u2)
    y = sin(2*pi*u2)
    z1 = r*x
    z2 = r*y

    return z1, z2
```

3.3 Choppy Waves

Choppy waves is a feature which adds irregularities to the ocean waves and break it's uniform shape as shown in Figure 5. Sharpness will be added to the wave's peaks and troughs. This feature enables the simulation to look more realistic and believable as if winds and weather was present. This is done by adding x to its new coordinates at time t by defining $x = x + \lambda D(x, t)$. As shown in Figure 6, we need two additional displacements, D_x and D_y using Equation 9. By combining the displacements, the final displacement map is achieved.

The movement function is defined as:

$$\mathbf{D}(\mathbf{x}, t) = \sum_{\mathbf{k}} -i\frac{\mathbf{k}}{k}\ \tilde{h}(\mathbf{k}, t)\ \exp\left(i\mathbf{k}\cdot\mathbf{x}\right) \qquad (9)$$

The only thing which was adjusted compared to the vertical height field generation, is the elements of the height amplitude Fourier components $\tilde{h}(k, t)$ were multiplied by the factor $-i\frac{k}{k}$.

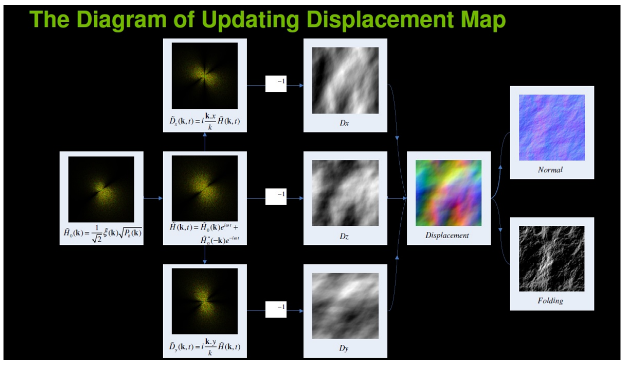
The Diagram of Updating Displacement Map
$\tilde{D}_x(\mathbf{k},t) = i\frac{\mathbf{k}.x}{k}\tilde{H}(\mathbf{k},t)$
$\tilde{H}_0(\mathbf{k}) = \frac{1}{\sqrt{2}}\xi(\mathbf{k})\sqrt{P_h(\mathbf{k})}$
$\tilde{H}(\mathbf{k},t) = \tilde{H}_0(\mathbf{k})e^{i\omega t} + \tilde{H}_0^*(-\mathbf{k})e^{-i\omega t}$
$\tilde{D}_y(\mathbf{k},t) = i\frac{\mathbf{k}.y}{k}\tilde{H}(\mathbf{k},t)$
-1
-1
-1
Dx
Dz
Dy
Displacement
Normal
Folding

Figure 6: *Updating Displacement Map*

Figure 7: *Photograph of a real ocean.*

4 Evaluation

Compared to the Gerstner Wave implementation in Figure 2, the implemented FFT Ocean in Unity Engine as shown in Figure 8 is more realistic and looks much more similar to the real ocean as shown in Figure 7.

As mentioned by Wang et al., 2011, there are issues with choppy waves, especially when we increase the lambda value. This leads to a problem where the choppy wave simulation experiences "overlapping." Essentially, the external force is too strong for the waves to maintain their intended shapes, resulting in a visible effect as shown in 9. Setting it too low will result in underlapping in the centre. Thus, the lambda value (λ) is set at -2.

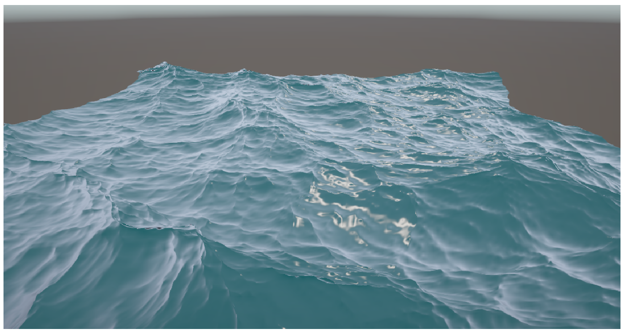

Figure 8: *Implemented Ocean in Unity.*

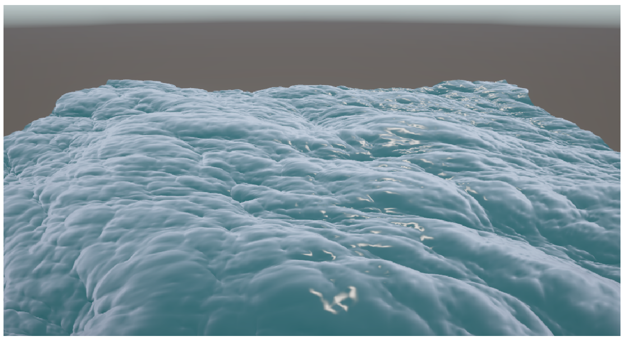

Figure 9: *Lambda Value (λ) set at 2.*

Performance is smooth at 800 FPS at 1080p with a Nvidia RTX 3070 GPU.

4.1 Further Works

The study can be taken further by including foam and spray detection to the simulation. Tessendorf et al., 2001 mentions using the Jacobian of the displacement map to calculate a convenient map that will only mask the wave peaks. This is done using the following Equations:

$$\begin{aligned}
&J(\mathbf{x}) = J_{xx}J_{yy} - J_{xy}J_{yx} \\
&J_{yy}(\mathbf{x}) = 1 + \lambda\frac{\partial D_y(\mathbf{x})}{\partial y} \\
&J_{xx}(\mathbf{x}) = 1 + \lambda\frac{\partial D_x(\mathbf{x})}{\partial y} \\
&J_{xy}(\mathbf{x}) = 1 + \lambda\frac{\partial D_x(\mathbf{x})}{\partial y} = J_{yx}(\mathbf{x})
\end{aligned} \tag{10}$$

The map will be considered as the folding map as it represents where the wave peaks self intersect until they fold on themselves.

$D = (D_x, D_y)$ is the function of the coordinate (D_x, D_y) on the horizontal plane. The Jacobian is less than zero if the $\mathbf{x}$ is in the overlapping region. As elaborated by Wang et al., 2011, the Jacobian equals zero in cases where some points will be transformed to the same location, indicating that the transformation is not reversible. When the Jacobian is less than zero, there is an area of overlap. Consequently, the Jacobian can be used to determine the positions of the overlapping region, masking them and adding a foam colour.

5 Conclusion

To sum up, the Fast Fourier Transform (FFT) is a potent algorithm that can create lifelike ocean waves in real-time. When combined with Unity engine, it provides an ideal platform for implementing the FFT ocean in games and simulations. The FFT ocean technique has numerous benefits, including better performance, increased realism, and the ability to adjust wind and other parameters on the fly.

Overall, using the FFT ocean in Unity engine is an incredibly effective method for generating realistic ocean waves in games and simulations. Its ability to create intricate and dynamic wave patterns has the potential to improve the immersion and realism of ocean-based virtual environments. With ongoing advancements in technology and algorithms like FFT, the future of ocean simulations in games and virtual reality is promising.

Bibliography

Dialid (Nov. 2019). *Simulating normal random variables*. URL: `https://quantgirl.blog/box-muller-and-marsaglia-bray/`.

Flügge, Fynn-Jorin (2017). *Realtime GPGPU FFT ocean water simulation*. Tech. rep.

Gerstner, Franz (1809). "Theorie der wellen". In: *Annalen der Physik* 32.8, pp. 412–445.

Heideman, M., D. Johnson, and C. Burrus (1984). "Gauss and the history of the fast fourier transform". In: *IEEE ASSP Magazine* 1.4, pp. 14–21. DOI: `10.1109/MASSP.1984.1162257`.

Nvidia (2015). *Nvidia*. URL: `https://developer.download.nvidia.com/assets/gamedev/files/sdk/11/OceanCS_Slides.pdf`.

Suriano, Fabio (2017). *An introduction to realistic ocean rendering through FFT*. URL: `https://www.slideshare.net/Codemotion/an-introduction-to-realistic-ocean-rendering-through-fft-fabio-suriano-codemotion-rome-2017`.

Tessendorf, Jerry et al. (2001). "Simulating ocean water". In: *Simulating nature: realistic and interactive techniques. SIGGRAPH* 1.2, p. 5.

Wang, Chin-Chih et al. (Jan. 2011). "Ocean wave simulation in real-time using GPU". In: pp. 419 –423. DOI: `10.1109/COMPSYM.2010.5685474`.

Williams, Hailey (Sept. 2019). *Tutorial: Ocean Shader with gerstner waves*. URL: `https://80.lv/articles/tutorial-ocean-shader-with-gerstner-waves/`.