Thesis for the Degree of
Master of Engineering

Sea Surface Rendering using Unity for Virtual Reality

Graduate School of Industry
University of Ulsan

Department of Global Smart IT Convergence

SHOKHZOD ALIMARDANOV SHAYMARDON UGLI

Sea Surface Rendering using Unity for Virtual Reality

Supervisor: Chong, UiPil

A Dissertation

Submitted to
The Graduate School of Industry, University of Ulsan
In partial fulfilment of the Requirements
For the Degree of

Master of Engineering

by

SHOKHZOD ALIMARDANOV SHAYMARDON UGLI

Department of Global Smart IT Convergence
Graduate School of Industry, University of Ulsan
Ulsan, Korea

January 2022

Sea Surface Rendering using Unity for Virtual Reality

This certifies that the dissertation of Shokhzod Alimardanov Shaymardon Ugli is approved.

_____

Committee Chair: Prof. Jo, Dongsik

_____

Committee Member: Prof. Park, Ju Chull

_____

Committee Member: Prof. Chong, UiPil

Department of Global Smart IT Convergence
Graduate School of Industry, University of Ulsan
Ulsan, Korea

December 2021

Acknowledgement

This project would not have been possible without the support of many people. Many thanks to my adviser, Prof. Chong, UiPil, who read my numerous revisions and helped make some sense of the confusion. Also, thanks to my committee members, prof. Jo, Dongsik and prof. Park, Ju Chull, who offered guidance and support.

Thanks to the University of Ulsan for awarding me a Dissertation Completion Fellowship, providing me with the financial means to complete this project. And finally, thanks to my family, and numerous friends who endured this long process with me, always offering support and love.

Abstract

Virtual Reality (VR), Augmented Reality (AR), and Metaverse are some of the most frequent words these days. Introduction of Web 3.0 is promising potentially big changes in everyday life of people by incorporating VR, AR and Metaverse. Virtual reality (VR) is fully imaginary or replica of the real world. Augmented Reality (AR) is the technology which enhances real world by superimposing virtual objects onto real objects. Metaverse includes both VR, and AR to create novel, interactive world. Once fully developed, meetings, workspaces, gaming, virtual tourism, etc., will all be integrated in Metaverse. In the research, we concern about creating such a virtual space where virtual tours are conducted to make the marine tourism more accessible for wider audience.

We focus our efforts on the animated ocean surface, for which we leverage a set of wave spectrum models from oceanographic research. We discuss the intricacies of said models, as well as their fitness for real-time rendering.

The findings of the study were applied to render the Sea Environment around Ulsan, South Korea. Additionally, some dolphin and whale models were added to the scene. The scene is set to simulate marine life and tourism. The Unity game engine is exploited to render the simulation and ease the development. The resulting scene were recorded and played on the whale watching tour boat on October 31, 2021, Ulsan, South Korea.

At the end of the demonstration, we conducted a survey. In the survey, virtual reality sea surfaces were recommended as an alternative to the real-world marine tourism. Around 80% percent of participants supported the development of such virtual spaces.

Contents

1. Introduction

Recent advancements in the technology have created novel experiences. Today people no longer need to travel long distances to see their friends or family members and talk. They no longer even need to be at the office to carry their professional duties. Students are studying online, and more contents are coming in near future. One of the most famous technologies with huge market capital and opportunities is Metaverse.

The metaverse is a digital reality that combines aspects of social media, online gaming, augmented reality (AR), virtual reality (VR), and cryptocurrencies to allow users to interact virtually. Augmented reality overlays visual elements, sound, and other sensory input onto real-world settings to enhance the user experience. In contrast, virtual reality is entirely virtual and enhances fictional realities.

As the Virtual Reality (VR) grows, it will create online spaces where user interactions are more multidimensional than current technology supports. Instead of just viewing digital content, users in VR will be able to immerse themselves in a space where the digital and physical worlds converge. One such case can be watching whales on the sea in the virtual world.

Marine tourism has always been one of the most popular types of tourism and entertainment. The value of watching and feeling the life under and on the water has never been underestimated. The water is undoubtedly the central part of this appealing tourism.

Even though most part of the Earth is covered with water, there are millions of people who has very little chance to see the sea and learn because of the geographic location of their home country and financial situations that are not sufficient to afford such an expensive overseas travel. Fortunately, cheap, and more accessible alternative solutions, for example VR, are arising. This study investigates these alternative solutions and water surface renderings in the development of virtual tourism.

1.1 Problem Statement

As we mentioned previously, marine tourism is simply not an accessible type of entertainment in some places in the world. People would need a lot of financial resources to go overseas just to

watch water washing beaches. To travel on the boat and just go deep inside the sea is a dream of many. Moreover, watching sea life, such as whales and dolphins, is not something frequent even for people living side by side with the sea.

Ulsan is one of the famous places to watch whales in Korea. Thousands of people visit Ulsan and get on the whale watching boats organized every weekend 5-8 months in a year. Despite its high frequency, watching whales on the sea is very rare experience. Many people pay for the boat even though they know that they would not see marine mammals.

Fluid simulation has never been a computationally easy feat for computers to render real time. To give the intuition of how computationally complex the fluid simulation (sea surface in this case) is, suppose we have N waves and M points on the surface of the water. We would need to compute $N \times M$ calculations at every frame because we need to calculate every wave at every point. When N and M are small numbers, it is a feasible task, but to get realistic looking ocean simulations we want N and M to be up to thousands or even more. The computational complexity would be $O(N \times M)$ which is very expensive process.

Realistic ocean surface simulation has been successfully achieved in movies like Titanic, Water World, etc. Jerry Tessendorf developed the method which was used to render realistic sea surfaces in said movies. [1] Unfortunately, the method purposed by Jerry Tessendorf cannot recommend an easy-to-use model. It requires specialized artists to tweak its parameter for getting visually plausible water simulations. The need to decrease the parameters to tweak is essential in spreading the method to wider developer audience.

In conclusion, there are three problems we want to tackle. The first is the creation of an accessible marine tourism. The second is the computational complexity of water rendering methods. The third is the complex parameters of the ocean rendering method of Jerry Tessendorf.

1.2 Purpose Statement

This research concerns with finding alternative solutions to the problems stated above. The perfect solution in the case of marine tourism would be an accessible and approachable way of experiencing marine life. The solution should expand the marine tourism so that it can cover more people around the world while suggesting quality services. For the second problem, we would like to make the simulation with better and more efficient way so that computers can render at real-time

without breaking the simulation. Lastly, the simulation should leave as few parameters as possible for the end artist to tweak while producing the same realism as the method developed by Jerry Tessendorf.

1.3 Contributions

We made two primary contributions. The first is to employ different spectra algorithm that the one utilized by Jerry Tessendorf. Also, we provided the demonstration of the implementation in Unity. The second is development of a virtual tour space for whale watching tourism.

For the first contribution, we replaced the Phillips spectra used by Jerry Tessendorf with JONSWAP spectra model. By leveraging JONSWAP spectra, we could reduce the number of parameters that an end artist uses to generate realistic looking shaders. Although the spectra model we used was developed before this research, we supplied with thorough discussions, explanations, and implementations of them in Unity. Since Unity does not provide built-in water system, developers usually need to program their own. Without good understanding of mathematics behind the water simulations, graphics card programming, and shaders which are themselves very complicated topics, developers find it very intimidating and time consuming to have their own water system. We explained all the algorithms and highlighted the simulation data flow in Unity.

Second contribution is specializing this research towards marine tourism. We developed a virtual space where people can experience whale watching tours in virtual world. Additionally, we simulated the sea surface around Ulsan.

1.4 List of Publications

1. S. Alimardanov, U. Chong.: *Implementation of Audio Augmented Reality using Unity for Whale Images*. KICSP, South Korea, 2020.
2. S. Alimardanov, U. Chong.: *Audio Augmented Reality Using Unity for Marine Tourism*. IHCI 2020, Springer Nature Switzerland AG, 2021.
3. S. Alimardanov, U. Chong.: *Modelling Whale Skeleton using Autodesk 3DS Mask*. KICSP, South Korea, 2021.
4. S. Alimardanov, U. Chong.: *Implementation of Virtual Sea Environment by Various Wind Speeds*. KICSP, South Korea, 2021.

2. Background

In this chapter, we introduce the tools we used for this research such as Unity, and Virtual Reality (VR). Additionally, introduction of the wave spectra model that we intent to use as a main algorithm will be discussed.

The chapter is organized as follows: Section 2.1 will give a brief introduction to the main game engine used to simulate the virtual space. The next Section 2.2 defines the term Virtual Reality. Section 2.4 gives nomenclature and basic relationship that are used throughout this chapter. Section 2.5 explains the wave dispersion. Section 2.6 presents the wave spectrum concept, whereas Section 2.7 discusses the directional spreading. The last Section 2.8 returns the stationary spectrum to calculate amplitude of the wave.

2.1 Unity

Unity3D is a cross-platform game creation system developed by Unity Technologies containing game engine and an integrated development environment (IDE). It is used to develop games for websites, desktops, handheld devices and consoles. Previously released on Mac OS in 2005, it was then expanded to target more than fifteen platforms. It can be run on Windows or Mac OS X system. Supported Platforms include BlackBerry 10, Windows Phone 8, Windows, OS X, Linux (mainly Ubuntu), Android, iOS, Unity Web Player (including Facebook), Adobe Flash, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360, Xbox One, Wii U, and Wii. It includes an asset server and Nvidia PhysX physics engine (Cross-platform Application Development using Unity Game Engine). [2]

2.2 Virtual Reality

Virtual Reality (VR) is popular name for an absorbing, interactive, Computer-mediated experience in which person perceives a synthetic (simulated) environment by means of special human-computer interface Equipment. It interacts with simulated objects in that environment as if they were real. Several persons can see one another and interact in shared Synthetic environment such as battlefield.[3]

Virtual Reality is a term used to describe a computer-generated virtual environment that may be moved through and manipulated by a user in real time. A virtual environment may be displayed on a head-mounted display, a computer monitor, or a large projection screen. Head and hand tracking

systems are employed to enable the user to observe, move around, and manipulate the virtual environment.

The main difference between VR systems and traditional media (such as radio, television) lies in three dimensionality of Virtual Reality structure. Immersion, presence, and interactivity are peculiar features of Virtual reality that draw it away from other representational technologies. Virtual reality does not imitate real reality, nor does it have a representational function. Human beings have inability to distinguish between perception, hallucination, and illusions.[3]

2.3 Forces on the Sea and Waves

Ocean surface waves are generated by forces, such as storms, earthquakes, the gravity of sun and moon, but with wind being the most prominent one. Each of these forces influences one or more frequency bands of the electromagnetic wave spectrum. The combination of said frequency bands constitutes the spectrum of ocean waves.

	Period Band Range (s)		Frequency Band Range (Hz)	
Wave Type	Start	End	Start	End
Capillary	0	1×10^{-1}	∞	1×10^{1}
Ultragravity	1×10^{-1}	1×10^{0}	1×10^{1}	1×10^{0}
Gravity	1×10^{0}	3×10^{1}	1×10^{0}	3.33×10^{-2}
Infragravity	3×10^{1}	3×10^{2}	3.33×10^{-2}	3.33×10^{-3}
Long Period	3×10^{2}	4.32×10^{4}	3.33×10^{-3}	2.32×10^{-5}
Tidal	4.32×10^{4}	8.64×10^{4}	2.32×10^{-5}	1.16×10^{-5}
Transtidal	8.64×10^{4}	∞	1.16×10^{-5}	0

Table 2.1 A classification of ocean surface waves by period and frequency. [4]

band as well as the corresponding frequency band. The shortest period waves are the capillary waves, with periods less than 0.1s. These are the first waves which one is able to notice when the wind starts to blow on the ocean surface, they appear like a fine structure of small ripples of nearly capillary dimension. Next are the ultragravity waves with periods ranging from 0.1s to 1s. The ordinary gravity waves observed at sea have periods varying from 1s to 30s and are composed of two types of waves, namely sea waves and swell. Sea waves are gravity waves directly generated and affected by local winds, and after the wind ceases to blow, they are called swell. More generally, swell consists of wind-generated gravity waves that are not significantly affected by the local wind; therefore, they have been generated elsewhere or at some time in the past. Beyond the

ordinary gravity waves there are infragravity waves, with periods between 30 s and 5 min. Then there are the long-period waves with periods in the range 5 min to 12 h. Their representatives in the sea are tsunamis and storm surges. The ordinary tidal waves represent the astronomical tides with fixed wave periods of about 12 h and 24 h. At the end of the spectrum there are the transtidal waves with periods greater than 24 h. These include the longer period components of astronomical tides.

Ocean surface waves are also classified by their major governing forces. First, the disturbing force which causes the actual water displacement, and second, the restoring force which brings the water back to its position at rest. Winds are responsible for the generation of capillary waves, ultra-gravity waves, gravity waves and infragravity waves, whereas long-period waves like tsunamis are generated by storms and earthquakes. The ordinary tides are caused by the

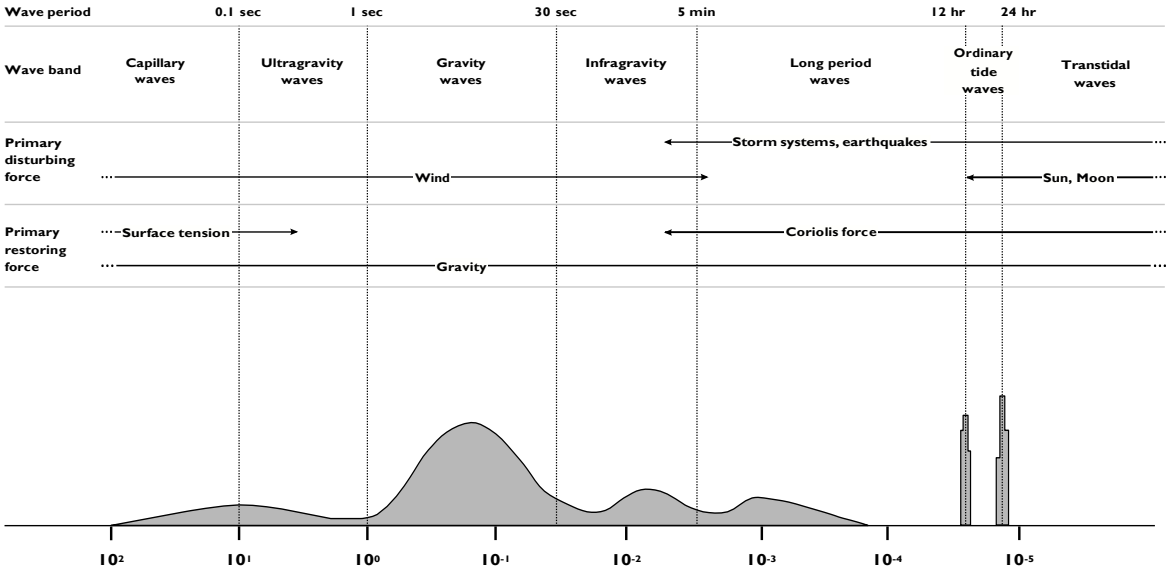

Figure 2.1 Wave frequency band, as well as the relative amount of energy. [4]

gravitational pull of sun and moon, transtidal waves, on the other hand, by storms, sun and moon. As soon as the water has been displaced, the force to bring capillary waves back to their position at rest is exerted by surface tension. Ultragravity waves, gravity waves and infragravity waves have gravity as their major restoring force. Gravity together with Coriolis act as the restoring force for long-period waves, ordinary tidal waves and transtidal waves. Figure 2.1 shows the entire spectrum of ocean surface waves and the forces responsible for various portions of the spectrum. As one can see, gravity is the main restoring force, therefore ocean surface waves are classified as surface gravity waves. Most of the energy of the spectrum is contained in two of the period ranges: the

ordinary gravity waves, and the ordinary tidal waves. Ordinary tidal waves, infragravity waves, long period waves, as well as transtidal waves do not contribute significantly to the actual optical appearance of the ocean in deep water, therefore we may omit them. The remaining types of waves have wind as the common disturbing force and are therefore grouped under the term wind generated waves, or in short, wind waves. Wind waves still contain a large part of the energy of the spectrum and range from ripples with less than a centimeter in height to huge waves more than thirty meters high.

Symbol	Meaning
λ	Wavelength
T	Period
ω	Angular Frequency
f	Ordinary Frequency
θ	Angle of wave relative to wind direction
$S(\omega, \theta)$	Wave spectrum
$S(\omega)$	Non-directional Wave Spectrum
$D(\theta)$	Directional Spreading Function
$\mathbf{k}$	Wave number vector, or Wave vector
k_x	X component of Wave vector
k_y	Y component of Wave vector
k	Magnitude of wave vector
$\omega = \varphi(k)$	Dispersion Relationship
E	Time average of wave energy
ρ	Density of water = 1000
σ	Surface Tension coefficient = 0.074
g	Gravitational constant = 9.81
h	Ocean depth
U	Average wind speed
F	Fetch
a	Wave amplitude
t	Time
ξ	Swell

Table 2.2 Nomenclature and Symbols

In order to be able to model the surface of such an intricate fluid system as the ocean, one has to reduce complexity. Research fields such as ocean engineering or coastal engineering employ the Airy wave theory to model the properties of the sea. Airy wave theory, also known as linear wave theory, describes the propagation of gravity waves on the surface of a homogeneous fluid with a set of linear equations. These equations give a decent approximation of the wave dynamics and kinematics with enough accuracy to model the state of the sea over a limited amount of time.

2.4 Nomenclature and Basic Relationships

The spectra presented in this thesis rely on a large number of physical parameters and different variables. Table 2.2 lists the symbols we use along with their meanings. The definition of some of these physical properties in terms of each other are given by the following definitive equations:

$$\omega = \frac{2\pi}{T} \tag{2.1}$$

$$\omega = 2\pi f \tag{2.2}$$

$$\mathbf{k} = \left(k_x, k_y\right) \tag{2.3}$$

$$k = \sqrt{k_x^2 + k_y^2} \tag{2.4}$$

$$k = \frac{2\pi}{\lambda} \tag{2.5}$$

$$\theta = \arctan\frac{k_y}{k_x} \tag{2.6}$$

2.5 Wave Dispersion

In order to discuss the wave spectra in detail, we must first present a functional relationship between the travel speed of waves and their wavelengths. This relationship is called the dispersion relationship, and is written as a function relating angular frequency ω to the wave number k. The dispersion relationships depend on gravity, ocean depth, and other physical parameters.

For deep water, in which the ocean depth is vastly larger than considered wavelengths, the dispersion relation- ship is derived as:

$$\omega^2 = gk \tag{2.7}$$

A finite-depth dispersion relationship is:

$$\omega^2 = gk \tanh kh \tag{2.8}$$

where *h* represents the ocean depth.

We can include a term which will only influence the relation- ship when waves have very small wavelengths, to simulate capillary wave dispersion. Capillary waves are small waves, typically with wavelengths less than a few centimeters, whose dynamics are primarily governed by surface tension. A dispersion relationship which incorporates the effects of surface tension at small scales as well as the full effects of gravity and ocean depth is:

$$\omega^2 = \left(gk + \frac{\sigma}{\rho}k^3\right) \tanh kh \tag{2.9}$$

where σ represents the surface tension coefficient in units of N/m, and ρ represents the density in kg/m^3. These relationships are all approximations, and there is a large set of different relationships to choose from.

The capillary formulation given in (2.9) is nearly identical to the finite depth formulation given in (2.8) for waves with large wavelengths, for which the $\frac{\sigma}{\rho}k^3$ term approaches zero. Similarly, the finite depth formulation is nearly identical to the deep-water formulation in (2.7) for large wavelengths when *tanh kh* approaches unity. Therefore, we always use the capillary formulation in our implementation. For mathematical analysis, though, we may resort to either the Deep or Finite Depth representations as the need for simplification demands.

2.6 The Wave Spectrum

The wave spectral density function, or wave spectrum, is a function which expresses the time-average energy of a random ocean configuration as a continuous function of angular frequency ω and direction θ. Spectral analysis of a height field represents the field as an infinite sum of discrete 2d cosine waves. [5]

A single 2D wave in an XYZ cartesian coordinate system, with Z up, can be described by the following equation:

$$\eta_i(x, y, t) = a_i \cos\left[k_i(\cos\ \theta_i + \sin\ \theta_i) - \omega_i t + \epsilon_i\right] \tag{2.10}$$

where a is an amplitude, ω is the angular frequency of the wave, k is the wavenumber, θ is the direction of the wave counterclockwise from the positive x-axis, t is time, and ϵ is the phase offset. Using vector notation,

$$\begin{aligned} \mathbf{r} &= (x, y) \\ \mathbf{k}_i &= (k_i \cos\,\theta, k_i \sin\,\theta) \\ &= \left(k_{xi}, k_{y_i}\right) \end{aligned}$$

$$\eta_i(\mathbf{r}, t) = a_i \cos\,(\mathbf{k}_i.\mathbf{r} - \omega_i t + \epsilon_i) \tag{2.11}$$

Because ω and θ are both functions of $\mathbf{k}$, a single wave is characterized entirely by the numbers $(\mathbf{k}_i, a_i, \epsilon_i)$. The entire ocean can be represented as just a sum of infinitely many of these $2D$ waves:

$$\eta(\mathbf{r}, t) = \sum_{i=0}^{\infty} a_i \cos\,(\mathbf{k}_i \cdot \mathbf{r} - \omega_i t + \epsilon_i) \tag{2.12}$$

We can formally define the wave spectrum in a useful form that will allow us to relate the spectrum to amplitudes in spectral space [5] :

$$\frac{a_i^2}{2} \approx S(\omega_i, \theta_i)\Delta\omega_i\Delta\theta_i \tag{2.13}$$

Each distinct random ocean surface has one and only one spectrum associated with it, and just as each surface is unique, each spectrum is unique. The different spectral models we discuss below in detail are intended to represent the average over a very large number of unique random oceans with the same physical characteristics. [6]

The directional wave spectrum $S(\omega, \theta)$ is written as the product of a non-directional spectrum $S(\omega)$ and a directional spreading function $D(\omega, \theta)$:

$$S(\omega, \theta) = S(\omega)D(\omega, \theta) \tag{2.14}$$

We present independent models for each component, starting with the non-directional spectra.

2.6.1 The JONSWAP Spectrum

Based on analysis of data collected as part of the Joint North Sea Wave Observation Project (JONSWAP), Hasselmann [7] that the oceans never reach a fully developed state. Nonlinear

interactions between waves, breaking of large wave crests, directional diffusion of wave energy, and turbulence in the wind itself all contribute to the continued development of waves. [5]

Hence an extra and somewhat artificial factor was added to the Pierson-Moskowitz spectrum [8] In order to improve the fit to their measurements. The JONSWAP spectrum is thus a Pierson-Moskowitz spectrum multiplied by an extra peak enhancement factor γ^r .The JONSWAP spectrum is formulated as follows:

$$S_{JONSWAP}(\omega) = \frac{\alpha g^2}{\omega^5}\exp\left(-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right)\gamma^r \tag{2.15}$$

$$\begin{aligned} r &= \exp -\frac{\left(\omega-\omega_p\right)^2}{2\sigma^2\omega_p^2} \\ \alpha &= 0.076\left(\frac{U^2}{Fg}\right)^0 .22 \\ \omega_p &= 22\left(\frac{g^2}{UF}\right) \\ \gamma &= 3.3 \\ \sigma &= \begin{cases} 0.07 & \omega \le \omega_p \\ 0.09 & \omega > \omega_p \end{cases} \end{aligned}$$

where *F* is the fetch, which is the length of the area over which the wind is acting on the water, formally defined as “the distance, from a lee shore”. The fetch term can be intuitively understood as the abstract size of the wind event over distance and time that is producing the waves. Young [9] presents an extremely detailed look at the measurement, geometry, and effects of fetch on wave spectra.

The JONSWAP Spectrum is similar to the Pierson Moskowitz spectrum, but waves continue to grow as distance and time grow (fetch), with a much more pronounced peak. Aesthetically, this means that the primary waves are strongly visible, with other wavelengths relatively damped out. This produces a much better visual match to photographs of ocean states.

2.6.2 The Texel MARSEN ARSLOE (TMA) Spectrum

The JONSWAP Spectrum [10] was fit to observations of waves in deep water. Kitaigorodskii [11] studied how deep-water spectra could be adapted to match field observations for shallow water

equilibrium ranges. They developed a frequency dependent function of ocean depth which can be applied as a multiplier to a deep-water spectrum. This is known as the Kitaigorodskii Depth Attenuation Function, and takes the form:

$$\Phi(\omega_h) = \left[\frac{(k(\omega,h))^{-3} \frac{\partial k(\omega,h)}{\partial \omega}}{(k(\omega,\infty)))^{-3} \frac{\partial k(\omega\infty)}{\partial \omega}} \right] \tag{2.16}$$

where ω_h is a dimensionless frequency defined by $\omega\sqrt{h/g}$. This is an intimidating function at first glance, but when plotted as a function of ω_h, as seen in Figure 2.2, we see that the function takes a very simple shape that we can easily approximate. An approximation for

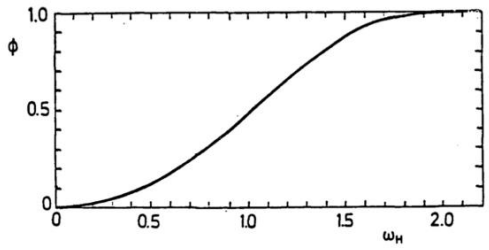

Figure 2.2 $\Phi(\omega,h)$ as a function of ω_h [5]

$\Phi(\omega,h)$ given in Thompson and Vincent [12] is:

$$\Phi(\omega,h) \approx \begin{cases} \frac{1}{2}\omega_h^2 & \text{if } \omega_h \leq 1 \\ 1 - \frac{1}{2}(2-\omega_h)^2 & \text{if } \omega_h > 1 \end{cases}$$

This approximation has less than 4% error over the plotted domain Hughes [10] . A smooth step function from 0 to 2.2 or a stretched and translated *tanh* function will also work.

Wonderfully, for all the detail and care taken, the final form of the spectrum is just the product of the JONSWAP spectrum and the Kitaigorodskii Depth Attenuation function:

$$S_{\text{TMA}}(\omega) = S_{\text{JONSWAP}}(\omega)\Phi(\omega,h) \tag{2.17}$$

The incorporation of depth damping alleviates a significant challenge for artists working to create plausible oceans in an environment.

2.7 Directional Spreading

The study of how wave energy disperses directionally, relative to the primary wind direction, is extremely elaborate. From a theoretical perspective, the amount of spreading is a function of energy concentration in each wavelength, which is related to the notion of the sea's development, and is therefore a function of the primary, wind speed, the fetch, and the spectrum. Furthermore, there is a distinction between the directional spreading of the locally affected Sea versus the directional spreading of the Swell, which is wave, energy that travels into an area from excitation by wind or storm elsewhere. [13] [5]

The Directional Spreading function is written $D(\omega, \theta)$. It multiplies the non-directional spectrum function $S(\omega)$ to produce the directional spectrum $S(\omega, \theta)$, as shown in equation (2.14). Any form of $D(\omega, \theta)$ must satisfy the condition:

$$\int_{-\pi}^{\pi} D(\omega, \theta) \mathrm{d}\theta = 1 \tag{2.18}$$

2.7.1 Mitsuyasu Directional Spreading

Mitsuyasu [14] fit a directional data set captured with a cloverleaf buoy to a function with a single shaping parameter, s, of a form:

$$D(\omega, \theta) = Q(s)|\cos(\theta/2)|^{2s} \tag{2.19}$$

where $Q(s)$ is a normalization factor to satisfy the condition in (2.18). Ochi [13] provides a closed-form approximate solution for $Q(s)$ expressed in terms of the Euler gamma function:

$$Q(s) = \frac{2^{2x-1}}{\pi} \frac{\Gamma(s+1)^2}{\Gamma(2s+1)} \tag{2.20}$$

The parameter s is fit to the data as follows:

$$s = \begin{cases} s_p(\omega/\omega_p)^5 & \omega \leq \omega_p \\ s_p(\omega/\omega_p)^{-2.5} & \omega > \omega_p \end{cases} \tag{2.21}$$

where ω_P is the peak angular frequency for the non-directional spectrum, and s_p is calculated from ω_p and physical parameters as follows:

$$s_p = 11.5\left(\omega_p U/g\right)^{-2.5} \tag{2.22}$$

2.7.2 Hasselmann Directional Spreading

Hasselmann [7] fit the JONSWAP data set to produce a directional spreading function that uses form in Equation (2.19), but with *s* computed differently:

$$s = \begin{cases} 6.97\left(\omega/\omega_p\right)^{4.06} & \omega \le \omega_p \\ 9.77\left(\omega/\omega_p\right)^{-2.33-1.45\left(\left(U\omega_p/g\right)-1.17\right)} & \omega > \omega_p \end{cases} \tag{2.23}$$

2.8 Initial Wave Spectrum and Directional Spreading

The initial state consists of two 2D arrays of complex numbers, each of which is $(N/2)+1$ in width (in the x dimension) and N in height, where N represents the width and height of the desired spatial height field. In our system, we constrain N to be a power of two, though most FFT [15] software can handle arbitrary sizes. When iterating in spectral space, the iteration coordinate is over discrete values of the wave vector **k**. However, our spectra have all been written in terms of the variables (ω, θ). To change variables correctly, the change of variables theorem has to be employed. The said step gives the equation below:

$$S\left(k_x, k_y\right) = S(\omega, \theta)\frac{\mathrm{d}\omega}{\mathrm{d}k}/k \tag{2.24}$$

The relationship between wave amplitude and the wave spectrum was described in equation (2.13), which can be solved for the mean amplitude:

$$\bar{a}\left(k_x, k_y, \Delta k_x \Delta k_y\right) = \sqrt{2S\left(k_x, k_y\right)\Delta k_x \Delta k_y} \tag{2.25}$$

3. Implementation

The previous chapter discussed the wave spectrum concept, a theoretical framework developed by oceanographic research to describe the distribution of wave energy among ocean surface waves which have been exposed to identical conditions, such as wind speed and distance to shore. Based on wave spectra we are able to synthesize ocean surfaces of great variety: from a perfectly calm sea, over one slightly agitated by the wind, to one with massive waves caused by a storm. But to be able to visualize such diverse seas in a convincing manner we are dependent on more data than surface elevation alone. Therefore, as described by Tessendorf, [1] we may capitalize on the spectral representation of ocean waves to obtain three distinct additional datasets. One, high-quality normal vectors by deriving the ocean surface's slope vectors in frequency space. Two, displacements to transform the gentle form of the sea into a more agitated one, where the waves are more similar to Gerstner waves than to sinusoids. Three, the first-order partial derivatives of aforementioned displacements, as they allow us to deduce the locations on the ocean surface where foam and spray may arise. Each of the three datasets requires us to construct additional spectra, all derived directly from the ocean surface elevation spectrum.

Alongside data synthesis, it is the rendering of a water body as large as the ocean that poses a challenge. The variety of possible camera views may range from closeups of the water surface to vistas where the sea spans all the way to the horizon. In the former case we would like to be able to observe small-scale detail on the water surface, such as ripples caused by gravity-capillary waves. The latter case is not difficult to handle in principle, because the ocean surface data we synthesize is seamlessly tileable. Still, we would prefer that the viewer may not be able to notice that the ocean surface repeats itself in all directions. To tackle both issues at once, we adopt the level-of-detail approach taken by Eric Bruneton and Dupuy [16] and Bruneton. [17] We compute a set of ocean surface tiles of differing size, where each tile samples a different part of the wave energy spectrum. By combining the tiles, we may extend the sum of waves as far as into the gravity-capillary wave domain. As for the tiling artifacts, even though we are unable to entirely remove periodicity by combing multiple tiles, we are at least able to increase the period to the least common multiple of the different tile sizes.

As one may already guess, we are burdened with a significant number of Fourier Transforms, based on both the number of spectra per tile (surface elevation, slopes, displacements, first order

derivatives of the displacements), and the number of different tiles. Thus, we need to make sure to keep the computational workload on an acceptable level for real-time display. We chose to tackle the issue at hand in a twofold manner. First, key properties of the ocean surface spectrum allow us to accelerate the computation of its Inverse Fourier Transform considerably. Second, we reduce the amount of spectral data to transform. We achieve the latter by improving upon the level-of-detail approach by Eric Bruneton and Dupuy [16] and Bruneton [17] with a multi-resolution scheme, where specific datasets are generated at a lower resolution than the other ones.

The remainder of this chapter is organized as follows: Section 3.1 gives a compact summary of ocean surface synthesis by means of a wave energy spectrum. Section 3.2 discusses the equations of slopes and displacements. Section 3.3 shows the heigh level data flow of the ocean simulation in Unity. In addition, sub-section 3.3.1 is dedicated for the demonstration of the spectrum implementations in unity compute shaders.

3.1 Spectrum Synthesis

Using the spectra we discussed in the previous chapter, we defined surface elevation as follows:

$$\eta(\mathbf{x},t) = \sum_{\mathbf{k}} \frac{1}{\sqrt{2}}(\xi_r + \mathrm{i}\xi_i)\sqrt{2S(\omega)D(\theta,\omega)\frac{d\omega(\boldsymbol{k})}{dk}\frac{1}{k}\Delta k_x \Delta k_z}\ \mathrm{e}^{\mathrm{i}\omega(\mathbf{k})t}\mathrm{e}^{\mathrm{i}\mathbf{k}\mathbf{x}} \tag{3.1}$$

where ξ_r and ξ_i are random scalars drawn from the Standard Normal Distribution. Both the wavevector domain $\mathbf{k}$ and the spatial domain $\mathbf{x}$ are of resolution $N \times N$, where N is a natural number, and a power of two. The latter requirement is a concession to the Fast Fourier Transform algorithm, which works fastest at such resolutions. Moreover, the spatial domain x has an extent of $L \times L$, thus the grid point spacing of the wave vector domain is as follows:

$$\Delta k_x = \Delta k_z = \Delta k = \frac{2\pi}{L}$$

We may subsume part of Equation (3.1) into the following:

$$h_0(\boldsymbol{k}) = \frac{1}{\sqrt{2}}(\xi_1 + i\xi_2)\sqrt{2S(\omega)D(\theta,\omega)\frac{d\omega(\boldsymbol{k})}{dk}\frac{1}{k}\Delta k_x \Delta k_z} \tag{3.2}$$

where $h_0(\mathbf{k})$ represents a random generated spectrum based on wave energy spectrum. As long as the wavevector domain and the wave energy spectrum do not change, h_0 does not change either. Hence, it is necessary to generate h_0 only once for a specific set of parameters such as area, resolution, wind and fetch. As h_0 by itself has no notion of time, we still need to take care of animation. Based on Equation (3.1) and (3.2) we may write:

$$h(\mathbf{k}, t) = h_0(\mathbf{k})\mathrm{e}^{\mathrm{i}\omega(\mathbf{k})t} \tag{3.3}$$

3.1.1 Hermitian Wave Spectrum

The spectrum $h(\mathbf{k}, t)$ from Equation (3.3) is not Hermitian for two reasons. First, the real-valued part of the spectrum would need to be an even function, which is only the case if the directional spread employed by the wave energy spectrum is centrosymmetric. Only two of the directional spreading functions presented in this work are centrosymmetric: the Unified spectrum's and the Phillips spectrum. Second, each element of $h_0(\mathbf{k})$ is generated in combination with a pair of random numbers, which makes it close to impossible to end up with matching pairs $h_0(-\mathbf{k}) = h_0(\mathbf{k})^*$. As we would like to profit from the performance gains an optimized Inverse Fourier Transform for real-valued functions offers, we are in need of a Hermitian spectrum. Thus, we may rewrite Equation (3.3) as follows:

$$h(\mathbf{k}, t) = h_0(\mathbf{k})e^{i\omega(k)t} + h_0(-\mathbf{k})e^{-i\omega(k)t} \tag{3.4}$$

which gives us a Hermitian spectrum, independent of both the generated random numbers and whether the wave energy spectrum is an even function or not.

Note: Surface elevation η gives the same results for Equation (3.3) and Equation (3.4). Still, Equation (3.3) requires a standard Inverse Fourier Transform with a complete complex-valued source spectrum and a complex-valued result array of the same size. The real-valued part of the result array represents the surface elevation, whereas the imaginary valued part is a by-product which holds no relevant information and is therefore to be discarded.

3.2 Slopes and Displacements

At this point we can compute surface elevation. For lighting purposes, we also need to find the surface's slope vectors to be able to compute the surface's normal vectors. The simplest way to compute

the slope is through finite differences in the spatial domain of the surface. Such an approach is efficient memory- and computation-wise, but may lack quality, because it can be a poor approximation to the slope of waves with small wavelengths. We are able to obtain more precise slope vectors by employing spectral differentiation. Spectral differentiation allows us to find the derivatives of a function via the Fourier Transform.

First, we will look at the more simple, one-dimensional case of spectral differentiation. We reduce the spatial domain as well as the wavevector domain from beforehand to one dimension. Let N be the resolution of the spatial and wavenumber domains, and L the size of the spatial domain. Moreover, let $\alpha \in \mathbb{N}$ with $-\frac{N}{2} \leq \alpha < \frac{N}{2}$, let the spatial domain be $x \in \alpha \frac{L}{N}$, and let the wavenumber domain be $k \in \alpha \frac{2\pi}{L}$. Let *g(k)* be the Discrete Fourier Transform of *f (x)*, then we may write the Inverse Discrete Fourier Transform as follows:

$$f(x) = \sum_k g(k) \mathrm{e}^{\mathrm{i}kx}$$

If we follow the lead of the continuous Fourier Transform, then, in the discrete case, we may compute the n[th] derivative of *f (x)* as follows:

$$\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n} = \sum_k (\mathrm{i}k)^n g(k) \mathrm{e}^{\mathrm{i}kx} \tag{3.5}$$

Johnson [4] shows that Equation (3.5) is not correct for odd n in combination with even N. For example, if *f (x)* is a real function, then *g(k)* is Hermitian. But *ik g(k)* is not Hermitian for even N therefore the first order derivative of *f (x)* would end up being a complex-valued function instead of a real-valued one. To get correct results for all n and N, we rewrite the above equation as follows:

$$\frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n} = \sum_k d(k,n) g(k) \mathrm{e}^{\mathrm{i}kx} \tag{3.6}$$

$$d(k,n) = \begin{cases} 0 & \text{if } n \text{ odd, and } N \text{ even, and } |k| = \frac{N}{2}\frac{2\pi}{L}, \\ (\mathrm{i}k)^n & \text{else.} \end{cases} \tag{3.7}$$

Now we may go back to the two-dimensional case, with our original two-dimensional domains **x** and **k**. The resolution of said domains is $N \times N$, the size of the spatial domain is $L \times L$. Let *g(k)* be the Discrete Fourier Transform of *f (x)*, then we may write the Inverse Discrete Fourier Transform [18] in two dimensions as follows:

$$f(\mathbf{x}) = \sum_{\mathbf{k}} g(\mathbf{k}) e^{i\mathbf{kx}}$$

Based on Equation (3.6), and by reusing Equation (3.7) for both dimensions (N and L are equal for both dimensions), we find the derivatives of *f(x)*:

$$\frac{\partial^{n+m} f(\mathbf{x})}{\partial x^n \, \partial z^m} = \sum_{\mathbf{k}} d(k_x, n) d(k_z, m) g(\mathbf{k}) e^{i\mathbf{kx}} \tag{3.8}$$

where *x* and *z* denote the two components of vector **x**, and k_x and k_z the two components of wavevector k respectively. As we need to find the surface slope vector, we need to compute the surface elevation's first order partial derivatives. Given surface elevation η(**x**, t), we obtain the two-dimensional surface slope vector s as follows:

$$\mathbf{s}(\mathbf{x}, t) = \left[\frac{\partial \eta(\mathbf{x}, t)}{\partial x}, \frac{\partial \eta(\mathbf{x}, t)}{\partial z}\right] \tag{3.9}$$

We leave it as an exercise for the reader to substitute the terms η(**x**, t) and h(**k**, t) into Equation (3.8) to be able to compute the surface slope vector as given in Equation (3.9).

The two spectra which constitute the surface slope vector in the wavevector domain are Hermitian. Therefore, we are able to apply the optimization, which allows us to combine both spectra into one, obtaining both components of the surface slope vector with just one Inverse Fourier Transform.

3.2.1 Normal Vectors

In this work we employ a right-handed Cartesian coordinate system where the positive Y-axis points in the opposite direction of earth's gravity. The two-dimensional surface slope vector lies in the XZ-plane of the world space coordinate system. Based on the surface slope vector we may find the unit length surface normal vector n as follows:

$$\mathbf{n} = \frac{(-s_x, 1, -s_z)}{\|(-s_x, 1, -s_z)\|}$$

Where s_x and s_z denote the two components of slope vector **s** as obtained by Equation (3.9).

3.2.2 Displacements

The waves generated with the methods presented up to now tend to have rounded peaks and troughs, which is typical for fair weather conditions. But we also would like to be able to synthesize waves

matching poor weather conditions, such as strong winds or even storms. During such weather, ocean waves are sharply peaked at their tops and flattened at the bottoms. Tessendorf [1] describes a method based on the Fourier Transform to produce such choppy waves. The concept is simple: displace the grid points of the spatial domain in the XZ-plane, with the displacement varying locally with the waves. Note that the displacement is a two-dimensional vector, therefore we end up computing a vector field. Similar to spectral differentiation, we compute an Inverse Fourier Transform of the spectrum h, where the latter is modified by an additional term. We may write the displacement vector field as follows:

$$\boldsymbol{D}(\boldsymbol{x}, t) \;=\; [D_x(\boldsymbol{x}, t), D_z(\boldsymbol{x}, t)] \tag{3.10}$$

with

$$D_x(\mathbf{x}, t) = \sum_{\mathbf{k}} d(k_x, k) h(\mathbf{k}, t) \mathrm{e}^{\mathbf{ikx}} \tag{3.11}$$

$$D_z(\mathbf{x}, t) = \sum_{\mathbf{k}} d(k_z, k) h(\mathbf{k}, t) \mathrm{e}^{\mathbf{ikx}} \tag{3.12}$$

$$d(l, k) = \begin{cases} 0 & \text{if } N \text{ even, and } k = 0 \text{ or } |l| = \dfrac{N}{2}\dfrac{2\pi}{L} \\ -\mathrm{i}\dfrac{l}{k} & \text{else.} \end{cases} \tag{3.13}$$

where the term *d(l, k),* in addition to make sure the resulting spectrum is Hermitian, also avoids a division by zero at $k = 0$. Because the spectra on the righthand side of Equations (3.11) and (3.12) are Hermitian, we are again able to obtain both components of the displacement vector field with just one Inverse Fourier Transform.

Given grid point **x** = (*x*, *z*), time *t* and surface elevation η(**x**, t), we may write the corresponding three-dimensional, vertically displaced vertex coordinate as follows:

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} x \\ \eta(\mathbf{x}, t) \\ z \end{bmatrix} \tag{3.14}$$

With displacement vector **D**(**x**, t) at hand, according to Tessendorf (1999) we may compute the three-dimensional, vertically and horizontally displaced vertex coordinate as follows:

$$\mathbf{w}(\mathbf{x}, t) = \begin{bmatrix} x + uD_x(\mathbf{x}, t) \\ \eta(\mathbf{x}, t) \\ z + uD_z(\mathbf{x}, t) \end{bmatrix} \tag{3.15}$$

here $u \in \mathbb{R}$ is a user-controlled parameter to scale the importance of the displacement vector. We have found that the above formula is correct as long as $u \leq 0$, otherwise the effect is the opposite of what we want to achieve. Looking at Equation (3.15), one can see that it is not the wave heights that are altered, but instead the grid points on the XZ-plane are transformed based on the spatial structure of the

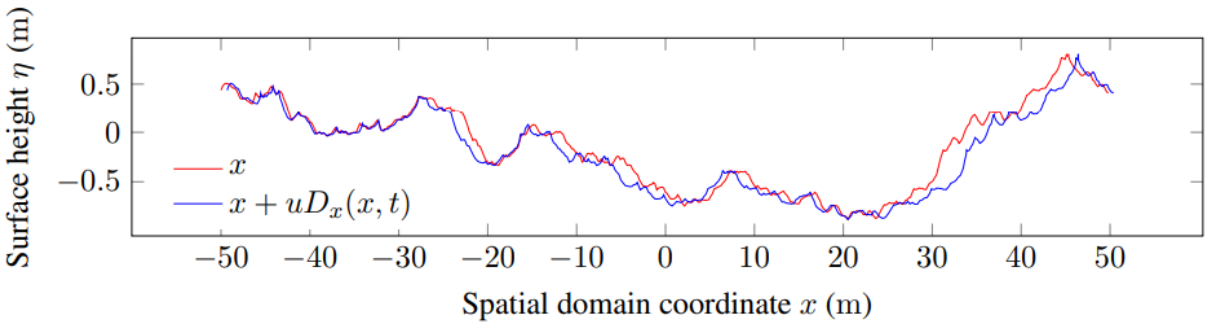

Figure 3.1: Red: the original wave profile. Blue: the displaced wave profile, note the step tops and the flattened walleys (*u = -3).* [4]

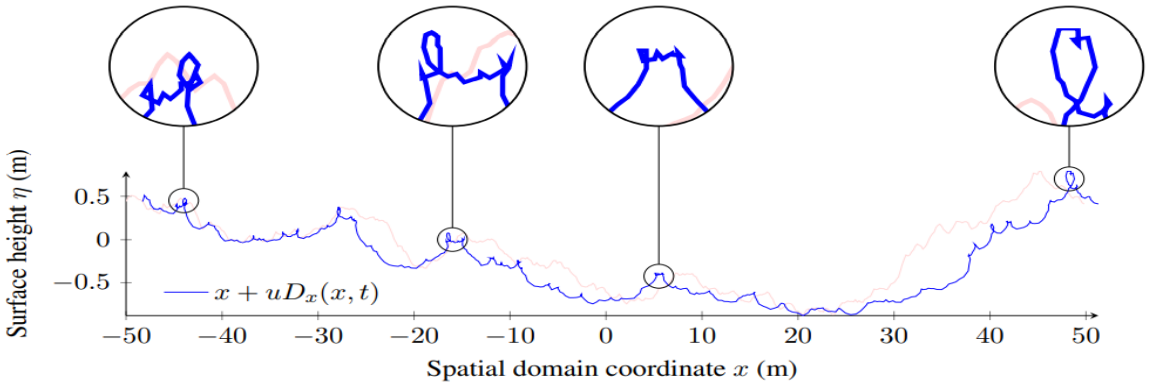

Figure 3.2: Background: the original wave profile. Foreground: the displaced wave profile, with some of the self-intersections highlighted. We chose a large displacement scaling parameter (u = -7.5) to better emphasize the effect of self-intersection. [4]

height field. This particular transformation accomplishes the effects we sought: the waves' peaks are sharpened, and the waves' valleys are broadened. Figure 3.1 shows two profiles of the wave height along one direction, where one profile uses displaced coordinates, while the other does not.

3.2.3 Slopes after Displacement

Based on Figure 3.1 one may see that the application of displacements does not only alter the underlying grid points, but as a direct consequence the surface's slope, too. Hence, we need to adjust our computation of the slope vector to the new circumstances. Given surface elevation η(**x**, t), displacement vector **D**(**x**, t), and grid point transformation **x** + u**D**(**x**, t), according to Dupuy and Bruneton , private communication] we obtain the two-dimensional surface slope s as follows:

$$\mathbf{s}(\mathbf{x}, t) = \left[\frac{\frac{\partial \eta(\mathbf{x}, t)}{\partial x}}{1 + u \frac{\partial D_x(\mathbf{x}, t)}{\partial x}}, \frac{\frac{\partial \eta(\mathbf{x}, t)}{\partial z}}{1 + u \frac{\partial D_z(\mathbf{x}, t)}{\partial z}} \right] \tag{3.16}$$

where u is the displacement scale parameter. Now, in addition to surface height η, displacement vector **D**, and the slopes of η, we also need to compute a partial derivative of D_x and D_z each. Again, we may compute the latter two by means of spectral differentiation, see Equation (3.8).

3.2.4 Self-Intersections

The method by Tessendorf we use to generate choppy waves is not devoid of drawbacks. As shown in Figure 3.2**,** some of the displacement vectors we compute may be large enough to cause the displaced geometry to self-intersect. Near the top of some waves the surface, actually, passes through itself, and as a consequence invert, with the surface normal pointing inwards instead of outwards. We may get rid of this undesired side-effect in a rather simple way: reduce the magnitude of displacement scaling parameter u. On the other hand, Tessendorf suggests that we use said self-intersections to our advantage, because they may be able to signal the breaking of waves, as well as the production of foam and spray. Hence, we need to be able to test for such self-intersections in an efficient manner. Tessendorf recommends computing the determinant of the Jacobian matrix of the transformation from grid point x to displaced grid point **x** + u**D**(**x**, t). Based on the determinant we may decide if self-intersection is taking place. First, we need the Jacobian matrix of the grid point transformation **g**(**x**, t) = **x** + u**D**(**x**, t). The Jacobian matrix is the matrix of all first order partial derivatives of a vector valued function. As grid point transformation

g maps from $\mathbb{R}^2$ to $\mathbb{R}^2$, the Jacobian matrix is 2×2 in size. Let **J** be the Jacobian matrix of transformation g, then we may write:

$$\mathbf{J}(\mathbf{x},t) = \begin{bmatrix} J_{xx} & J_{xz} \\ J_{zx} & J_{zz} \end{bmatrix} = \begin{bmatrix} 1 + u\dfrac{\partial D_x(\mathbf{x},t)}{\partial x} & u\dfrac{\partial D_x(\mathbf{x},t)}{\partial z} \\ u\dfrac{\partial D_z(\mathbf{x},t)}{\partial x} & 1 + u\dfrac{\partial D_z(\mathbf{x},t)}{\partial z} \end{bmatrix}$$

Next, we compute the determinant of the Jacobian matrix:

$$\det\left(\mathbf{J}(\mathbf{x},t)\right) = J_{xx}J_{zz} - J_{zx}J_{xz}$$

If the Jacobian determinant at **x** is positive, then **g** preserves orientation near **x**. If it is negative, **g** reverses orientation. We are interested in the latter case, because wherever the surface self-intersects, it actually folds back on itself via a loop, and a loop requires a reversal of orientation. One can see that wherever the surface self-intersects, the Jacobian determinant is negative. Figure 3.3 depicts a displaced surface, as well as the corresponding Jacobian determinants. One can see that wherever the surface self-intersects, the Jacobian determinants is negative.

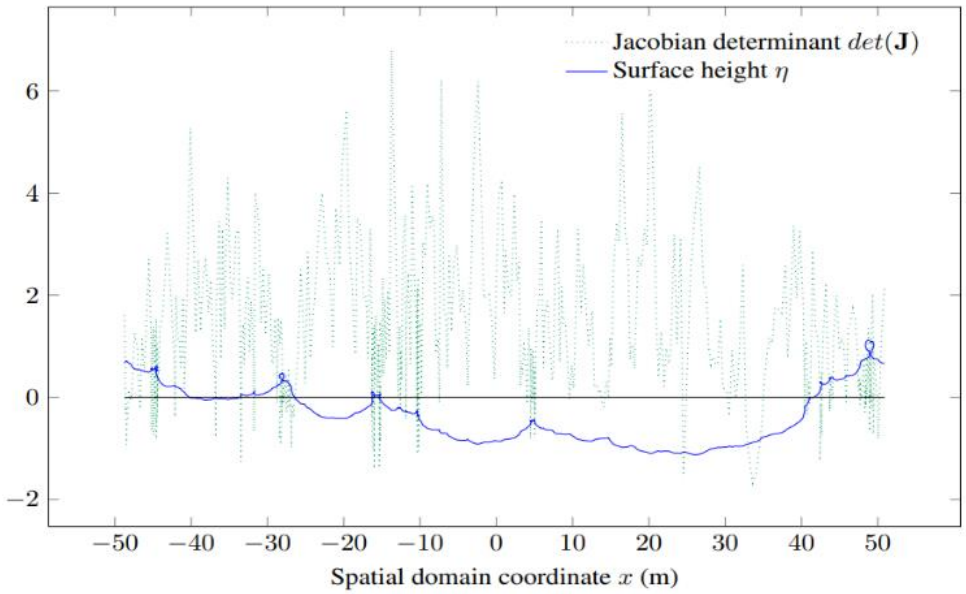

Figure 3.3: Blue: the displaced wave profile (u = 7.5). Green: the Jacobian determinant. Note, that each instance of surface self-intersection is accompanied by negative Jacobian determinants. [4]

3.3 Implementation in Unity

Finally, we implement the spectra and start simulation in Unity. Compute shaders, C# scripts and surface shaders are used for the rendering of the ocean surface (Figure 3.4).

Compute shaders are programs that run on the graphics card, outside of the normal rendering pipeline. They can be used for massively parallel GPGPU algorithms, or to accelerate parts of game rendering. Usually, compute shader files are written in HLSL (High Level Shader Language) and compiled or translated into all necessary platforms automatically. Thus, we use terms “Compute shader” and “HLSL” interchangeably. Compute shaders are responsible for calculation of the spectrum. In addition, we exploited an IFFT [18] implementation in compute shaders to delegate heaviest calculations away from CPU.

Firstly, compute shaders input initial data, such as wind speed, fetch etc. Secondly, it calculates the wave spectrum (Section 3.1.1), displacements (Section 3.2.2), slopes (Section 3.2.3). Up until this point everything calculated is in the frequency domain. At the third step, we convert the data from frequency domain to discrete time domain using IFFT.

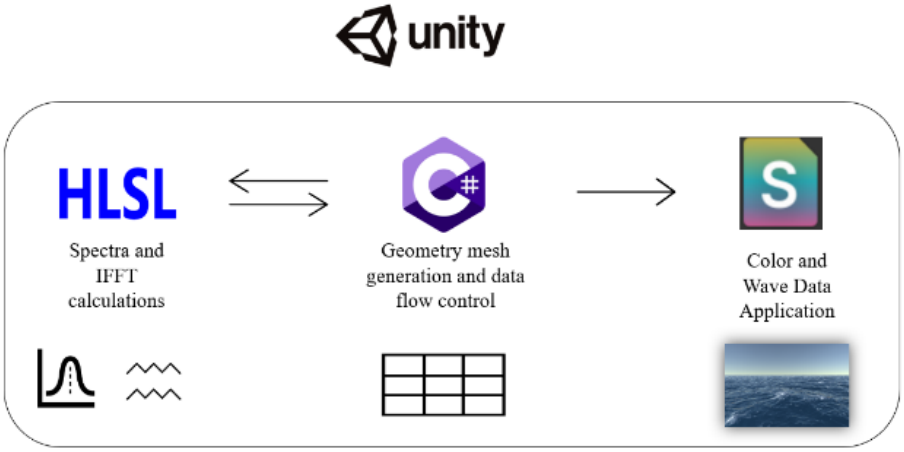

Figure 3.4 Implementation of the Sea Surface rendering in Unity Engine.

A Unity C# script is used to control the data flow. For example, it gets input from an artist (unity developer) and sends it to the HLSL program. When HLSL finishes the calculations, C# script retrieves the results. Meanwhile, we calculate the ocean geometry in C#. Since the spectrum gives the wave heights not the geometry itself, we need geometrical body to apply those values. Consequently, we generate the geometry in C#.

Next, C# sends the values of the spectrum as well as the surface geometry to a unity surface shader. Surface shaders, as the name implies, define the physical characteristics of Materials. They are responsible for both calculating the final color of each pixel within a Material and performing the light calculations that define the shading of each pixel on the surface. Most surface shaders in Unity are extensions of the default Standard Surface Shader, which makes the creation process more intuitive and allows artists to define the look of their surfaces more freely.

The data that unity surface shader inputs contains displacements, and derivatives of the displacements. In the shader, the displacements are used to modify the vertices of the ocean geometry. The derivatives of the displacements are used to calculate normals. Normals themselves are parameters that help the surface shader to apply colors and shadows.

Foam effects are also applied in the unity surface shader. However, the detection of the foam points is done using the determinant of Jacobian matrices as described in Section 3.2.4. When the determinant has a negative value, the shader applies the foam texture.

3.3.1 Spectrum Calculations in Compute Shaders

In this section, implementations of JONSWAP spectrum with TMA correction and Directional Spreading are demonstrated.

The first algorithm that HLSL calculates is the JONSWAP spectrum. Algorithm 1. shows the spectrum in HLSL code. In the "if-else" statement, surface tension coefficient, $\sigma\ (sigma), is$ chosen based the difference between $\omega\ and\ \omega_p$. The "r" variable at the following line is the power of peak enhancement factor, γ. Then, we put all variables together and calculate the JONSWAP spectrum. Note that TMA correction [19] is also applied here.

In Algorithm 2, the directional spreading function as shown in Section 2.7 is implemented. At the first line, a single shaping parameter, s, is calculated with additional swell parameter. C. Horvath [5] recommends including the swell parameter to generate visually plausible water simulation. Swell parameter is not defined in this research. An interested one can find thorough definition of the swell in the reference [5] At the second line of Algorithm 2, two directional spreading algorithms are linearly interpolated to produce mixed directional spreading. The first directional spreading function is the one which Jerry Tessendorf [1] is used. Again, the definition of

```
float JONSWAP (float omega, float g, float depth, SpectrumParameters pars)
{
	// Choosing sigma value based on omega and omega at peak
	float sigma;
	if (omega <= pars.peakOmega)
		sigma = 0.07;
	else
		sigma = 0.09;

	// Peak enhancement factor calculation
	float r = exp(- pow((omega - pars.peakOmega), 2)
		/ (2 * pow(sigma, 2) * pow(pars.peakOmega, 2)));

	// 1/W
	float oneOverOmega = 1 / omega;

	// Wp/W -- omega at peak divided by omega.
	float peakOmegaOverOmega = pars.peakOmega / omega;

	// JONSWAP spectrum algorithm
	float jonswapSpectrum = pars.scale
				* TMACorrection(omega, g, depth)
				* pars.alpha
				* pow(g, 2)
				* pow(oneOverOmega, 5)
				* exp(-1.25 * pow(peakOmegaOverOmega, 4))
				* pow(abs(pars.gamma), r);

	return jonswapSpectrum;
}
```

Algorithm 1. JONSWAP spectrum with TMA correction in Compute Shader code.

this algorithm is not included. In order to find further information, refer to [1] [5] "Cosine2s" method in the "lerp" function is the implementation of Equation (2.19). It inputs difference of theta and user specified angle, a single shaping parameter, s. Then, the "Cosine2s" method returns directional spreading value. The algorithm produces values which is linearly interpolated between methods of Tessendorf. [1] [5]

```
float DirectionSpectrum(float theta, float omega, SpectrumParameters pars)
{
	// Calculation of s, a single shaping parameter, using swell
	float s = SpreadPower(omega, pars.peakOmega) + 16
			* tanh(min(omega / pars.peakOmega, 20)) * pow(pars.swell, 2);

	// Linear interpolation between two directional spreading functions
	return lerp(2 / 3.1415 * pow(cos(theta), 2), Cosine2s(theta - pars.angle, s),
							pars.spreadBlend);
}
```

Algorithm 2. Directional Spreading in Compute Shader code.

At the end, JONSWAP spectrum and Directional Spreading functions are multiplied to get the initial spectrum of the water simulation.

4. Results

The Chapter 3 showed the implementation of the spectra in Unity compute shaders. In this chapter, rendering realistic Ulsan Sea environment will be discussed. The most important parameter in the sea simulation of this research is the wind speed. By altering it, we will be able to render any type of sea surfaces. As stated in chapter 2, this ocean simulation assumes the sea to be deep where the sea surface disturbing forces will not affect the water near underwater bed. Therefore, we will not simulate shallow water near beaches.

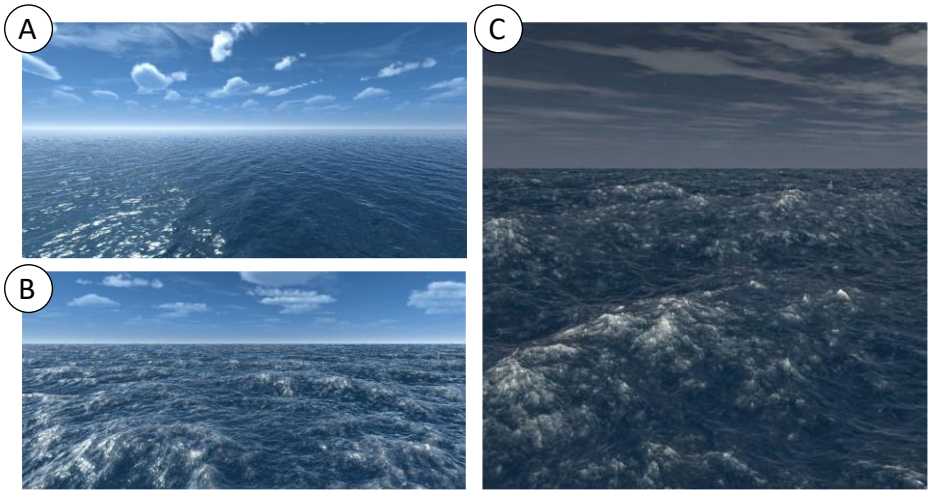

Figure 4.1: Implementation results. A) Calm sea surface 10m/s wind speed. B) Sea surface with medium wind speed (4m/s). C) Agitated sea surface of 10m/s wind speed.

To simulate deep Ulsan Sea surface, we have gathered high, medium, and small wind speeds. By applying the wind speed, realistic rendering of the sea surface is achieved. In addition to the sea rendering, we have also simulated marine mammals, such as dolphins and whales. The resulting scene then demonstrated on the whale watching tour boat to gather feedback from viewers.

This chapter concerns with showcasing implementation results. Section 4.1 discusses the sea rendering outputs where Section 4.2 presents the survey results collected on the whale watching tour ship.

4.1 Ulsan Sea Surface Rendering

The spectra model we have used in this rendering will be able to handle waves ranging from capillary waves to very agitated ones. The data taken from Ulsan Sea statistics shows that the highest wind speed on Ulsan Sea was around 10m/s on October 31, 2021.

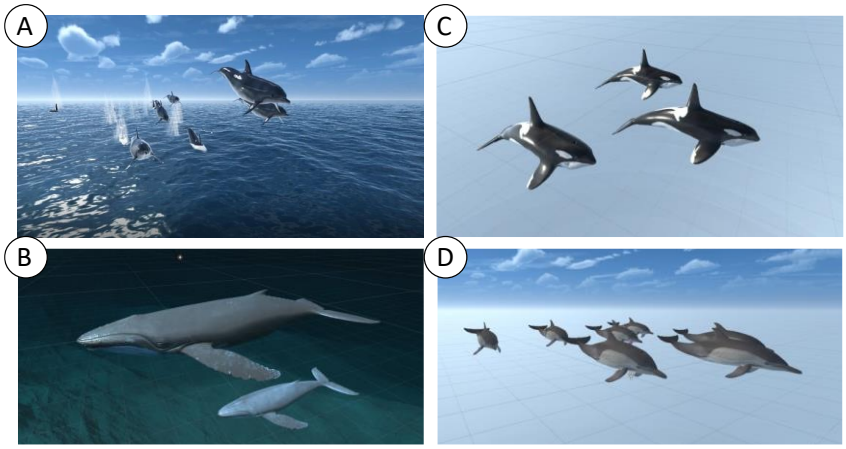

Figure 4.2: Additional 3D models added to the scene. A, D) are dolphin models. B, C) are whale models.

Figure 4.1 is the resulting sea surface rendering and Figure 4.2 is the 3D models which were used to create a virtual sea environment. To achieve as realistic as possible virtual water surface rendering, we need to tweak several parameters such as color, foam intensity, wind speed and more. Fortunately, many other parameters need to be set only once and it will match other weather conditions seamlessly. The only two things we must change are wind speed and foam. Figure 4.1 (A) shows very calm sea surface where wind speed is 0.1 and the foam intensity is zero. Figure 4.1 (B) renders the sea surface with the wind speed going up to 4m/s as well as an increased foam intensity. At this stage, the sea is calmer compared to the Figure 4.1(C). But the waves still have relatively high picks and stay agitated. Figure 4.1 (C) demonstrates the resulting agitated sea surface, where wind blows at the speed of 10m/s. The foam intensity is also modified to match the behavior of the waves.

4.2 Survey Results

On October 31, 2021, we have conducted a survey targeting the participants of Ulsan Whale watching tour ship, Table 4.1. The attendees were shown 3 types of videos where the different sea

surface states and animated dolphins were simulated. After video, participants were asked to fill survey sheets consisting of several questions. The questions were started with asking the age and gender of the visitors and, also, asked about the importance of Simulated Ulsan Sea environment in the customer satisfaction where watching whale on the real ocean has very low probability.

34 people participated in the survey of which 41% male and 55% female. The participants who took our survey answered 4 questions. The first question was about whether they hoped to see whales or dolphins on the sea before boarding on the ship. 21 people or 61% of them answered "Yes" meaning they had the intention of watching the marine mammals. Unfortunately, seeing whales on the sea are getting complicated human experience.

The second question was if they could not see whales on the sea in real life, would they watch them in the virtual. This time 29 (85%) positive answers were collected.

For the third question, we asked participant if the videos we had prepared that day was interesting and enjoyable to them. 88% participants wrote positive answers. The last question about their support for the development of this kind of products showed strong support reaching 84% of people.

№	Questions	"Yes"	"No"	"I don't know"
1	Did you ride a sightseeing boat thinking you could see a real whale?	68%	19%	13%
2	If you cannot see whales on the sea in real life, would you like to watch them in virtual reality?	91%	9%	0
3	Did the video you watched today helped you to understand whales and enjoy more?	88%	6%	6%
4	If these whale videos continue to be developed, would you like to take a tour boat again in the future?	84%	3%	13%

Table 4.1: The survey results.

In conclusion, the possibility of people watching whales on the sea in real life is decreasing. At the same time, people have the same amount of interest to explore marine life and getting on the boat with the intention of seeing them even though they know it might not happen. The virtual experience could be cheap and more accessible alternative solution to the problem. Indeed, the survey have shown interest and support of people.

5. Conclusions

We stated in the introduction that in this work we seek to synthesize animated ocean surfaces which are both believable and computationally feasible. To achieve said objective, we picked the choppy wave algorithm by Jerry Tessendorf, which adopts the wave spectrum concept from oceanographic research for computer graphics purposes. Tessendorf approximates the perturbations produced by the water surface by sampling the wave spectrum in frequency space via the Fast Fourier Transform algorithm. The FFT algorithm is highly efficient and generates a perturbation pattern which seamlessly tiles the ocean surface. An additional upside of the wave spectrum approach is that oceanographic research provides a wide range of wave spectra which are specifically tailored to the deep water of the open ocean. Said wave spectra are employed for oceanographic forecasts and simulations, therefore they give a high degree of realism. Thus, we chose to implement a wave-energy spectra that is well established in oceanographic literature.

The resulting sea surface is then presented to people on the tour boat to check if it would really be an alternative solution for difficulties described in the introduction. Several video clips showing the sea surface renderings with whales and dolphins in the scene were played on the whale watching tour boat in Ulsan. We have simulated different weather conditions. The disturbing farce was wind speed and restoring force was gravitational constant of the Earth. After the video, we have conducted a survey. The survey was about people expectations and their thoughts about the development of alternative, virtual tourism services to solve problems which are stopping people enjoy marine life. People overall showed high degree of interest. The positive answers constituted more than 80% of the results.

5.1 Limitations

This study has mostly concerned with simulating ocean surface where water depth was assumed as deep as the disturbing force on the surface do not affect the water near the bed. Hence, the simulations of waters near beaches are not possible with the current methodology. Another, limitation is certain weather conditions, such as typhoons, is not covered. The approach we have applied uses wind speed to generate waves. The wind has straight direction and it con not be modified to generated circular motion as it can be seen in typhoons.

The other limitation is the lack of water splashes. Although we have generated foam effect to look the rendering realistic, believable water splashes could not be achieved. Boat and whale wakes were another form of water splash which would drastically increase the realism of the simulation. Additionally, the computational complexity at the current state is not low enough to support smart phones and other computer that does not contain GPUs.

5.2 Future Works

The future research work is largely around solving the limitations of current research. Shallow water simulations, underwater simulations, water splashes and expanding the coverage of more diverse weather conditions are some of many. Additionally, the integration of blockchain technologies where people can buy and sell their digital assets and donate for marine tourism using crypto currencies as a form of payments are planned. In the future the realism of the ocean surface should be increased, and the computational complexity needs to be adjusted to be executed on various types of user gadgets, for example, smartphones.

References

[1] Jerry Tessendorf.: *Simulating Ocean Water*. In SIGGRAPH course notes. ACM, 1999.

[2] Unity Game Engine: https://unity.com. Last accessed 29.11.2021.

[3] Mandal S.: *Brief Introduction of Virtual Reality and its Challenges.* International Journal of Scientific and Engineering Research, 2013.

[4] Gamper T.: *Ocean Surface Generation and Rendering*. Master's thesis, 2018.

[5] Horvath Ch. J.: *Empirical Directional Wave Spectra for Computer Graphics*. Proceedings of the 2015 Symposium on Digital Production, 2015.

[6] Jocelyn Fréchot*.: Realistic Simulation of Ocean Surface Using Wave Spectra*. JVRB - Journal of Virtual Reality and Broadcasting, 2007.

[7] Hasselmann K., Barnett T. P., Bouws E., Carlson D. E., et al.: *Measurements of Wind-Wave Growth and Swell Decay During The Joint North Sea Wave Project (JONSWAP).* Deutsche Hydrographische Zeitschrift, 1973.

[8] Pierson W. J., Moskowitz L.: *A Proposed Spectral Form for Fully Developed Wind Seas Based on the Similarity Theory of S. A. Kitaigorodskii*. Journal of Geophysical Research, 1964.

[9] Young I.: *Wind Generated Ocean Waves*. Elsevier, 1999.

[10] Hughes S. A.: *The TMA Shallow-Water Spectrum Description and Applications*. Technical report, 1984.

[11] Kitaigorodskii, S., Krasitskii, V., Zaslavskii, M.: *On Phillips' Theory of Equilibrium Range In The Spectra of Wind-Generated Gravity Waves*. Physical Oceanography, 1975.

[12] Thompson E. F., Vincent C. L.: *Prediction of Wave Height In Shallow Water*. Proceedings of Coastal Structures 1983, American Society of Civil Engineers, 1983.

[13] Ochi M. K.: *Ocean Waves*. Cambridge University Press, 1998.

[14] Mitsuyasu H. E. A.: *Observations of The Directional Spectrum of Ocean Waves Using Clover-Leaf Buoy*. Journal of Physical Oceanography, 1975.

[15] Heideman M., Johnson D., Burrus C.: *Gauss and The History of The Fast Fourier Transform*. IEEE ASSP Magazine, 1984.

[16] Eric Bruneton, Fabrice Neyret, Nicolas Holzschuch.: *Real-Time Realistic Ocean Lighting Using Seamless Transitions from Geometry to BRDF*. Computer Graphics Forum, 2010.

[17] Jonathan Dupuy and Eric Bruneton.: *Real-Time Animation and Rendering of Ocean Whitecaps*. In SIGGRAPH Asia 2012 Technical Briefs, ACM, 2012.

[18] Cooley J. W., Tukey J. W.: *An Algorithm for The Machine Calculation of Complex Fourier Series*. Mathematics of Computation, 1965.

[19] Lee N., Baek N., Ryu K. W.: *A Real-Time Method for Ocean Surface Simulation Using The TMA Model*. International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), 2008.