TRABAJO ESPECIAL DE GRADO

SIMULACIÓN DE OLAS MARINAS USANDO TRANSFORMADA DE FOURIER

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Becerra LL., Josué
Para optar al Título
de Licenciado de Computación

Caracas, 2018

TRABAJO ESPECIAL DE GRADO

SIMULACIÓN DE OLAS MARINAS USANDO TRANSFORMADA DE FOURIER

TUTOR ACADÉMICO: Prof. Hector Navarro

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Becerra LL.,Josué
Para optar al Título
de Licenciado de Computación

Caracas, 2018

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Centro de Computación Gráfica

ACTA DEL VEREDICTO

Quienes suscriben miembros del jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el **Br. Josué Becerra, C.I. 20793760** titulado: **"Simulación de Olas Marinas usando Transformada De Fourier"** a los fines de optar por el título de Licenciado en Computación, dejan constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijó el día 14 de junio del 2019 a las 11:00 a.m., para que su autor lo defendiera en forma pública, en el Centro de Computación Gráfica, Facultad de Ciencias, mediante una presentación oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el Jurado decidió aprobarlo.

En fé de lo cual se levanta la presenta Acta, en Caracas a los catorce días del mes de junio del año dos mil diecinueve, dejándose también constancia de que actuó como Coordinador del Jurado el Prof. Hector Navarro.

prof. Robinson Rivas M.Sc.

Prof. Héctor Navarro, Tutor

REPUBLICA DE VENEZUELA
Facultad de Ciencias
Escuela de Computación
Universidad Central - Caracas

Francisco Moreno

Prof. Francisco Moreno, Jurado

Prof. Rhadamés Carmona, Jurado

Este trabajo está dedicado primeramente a Dios, a todas las personas quienes estuvieron conmigo durante mi travesía diaria en mi vida. A aquellos que me vieron crecer como persona, a mis familiares que se esmeraron arduamente, para que yo floreciera y en especial a la persona más importante para mí. Que gracias a Dios, hemos sido un gran equipo: Emeli María Francia Rivas Salazar.

Agradecimientos

Primero que nada quiero agradecer a Dios Todopoderoso, nuestro Adon Yahshua Ha Mashiaj por la oportunidad de formarme como profesional y como persona en esta casa de estudios. Y por haberme dado la sabiduría para llevar a cabo este trabajo. Quiero agradecer a la Universidad Central de Venezuela por los profesores, asesores y amigos que me acompañaron en este viaje, a mi tutor y profesor Héctor Navarro que aunque no esté con nosotros sacrificó una parte de su tiempo para evaluar este documento, y a los profesores del CCG por el apoyo técnico y científico para el desarrollo e implementación de este proyecto. También a la corporación Bimodal Technology por la colaboración que me ha dado avanzar, a pesar de diversos percances que han transcurrido durante la carrera.

Becerra Ll., Josué

SIMULACIÓN DE OLAS MARINAS USANDO TRANSFORMADA DE FOURIER

Tutor Académico: Héctor Navarro

Palabras Claves: Olas, Simulación FFT, Olas de Fourier.

Resumen: La animación 3D es uno de los desafíos más extensos en el mundo de la computación gráfica, ya que existen diversas técnicas con distintas estructuras y algoritmos que permiten que los modelos 3D cobren vida en el mundo digital. Por lo cual, son contemplados y estudiados en simulaciones, que son adaptadas en diversas disciplinas. Las olas marinas son uno de esos temas. El Presente Trabajo de Grado tiene como objetivo exponer una escena 3D, en donde se contempla las olas marinas, a través de la transformada de Fourier. Esta técnica tiene dos vertientes: la transformada discreta de Fourier y la transformada Rápida de Fourier, que son implementadas en un simulador, desarrollado en C++ utilizando glfw, glew y glm.

Índice

Lista de Figuras

Lista de tablas

INTRODUCCIÓN

La animación 3D es uno de los temas más amplios e importantes de la ciencia de la computación gráfica, ya que abarca muchas disciplinas en el área científica, y además en el área del cine y en los video juegos. Las olas marinas son parte del desafío y por lo tanto, es un tema que en el presente Trabajo Especial de Grado se va a exponer. Las olas marinas son simuladas usando diversas técnicas, entre los cuales está la Transformada Discreta de Fourier (DFT) y la Transformada Rápida de Fourier (FFT), que son métodos que consisten en representar diversas frecuencias de ondas. El propósito de este trabajo es presentar el esquema funcional del simulador de olas marinas, utilizando ambas técnicas. Esto involucra las fases de implementación, las pruebas desarrolladas a lo largo de la investigación, en conjunto con los resultados que se derivan de los mismos y las interpretaciones y conclusiones que se derivan de los mismos.

Además, se explican las funcionalidades de la aplicación y los parámetros escogidos para simular las olas marinas. Incluyendo las ilustraciones que visualizan el resultado de las pruebas ya realizadas.

Planteamiento del Trabajo

Simular el comportamiento de una ola no es una tarea trivial, sin embargo, existe un conjunto de técnicas que permite hacer simulaciones aproximadas a las olas reales. Estas técnicas se diferencian por el costo de procesamiento gráfico y consumo de memoria, por lo que diseñar una técnica de simulación y de propagación del oleaje es un tema abierto. Además incluyen ecuaciones matemáticas tales como las ecuaciones de Stokes, Modelos de Airy, Ecuaciones de Navier-Stokes, entre otras. Aunque, la precisión numérica es muy distinta y a la vez compleja.

JUSTIFICACIÓN

La importancia del estudio de las olas se centra en dos aspectos importantes: el enfoque científico y el enfoque práctico. En el enfoque científico, se encuentra la transferencia de la energía entre el océano y la atmósfera, abarcando áreas del conocimiento que competen con la meteorología y climatología. Desde el enfoque práctico se centra en los campos de la ingeniería civil y naval, como por ejemplo el diseño de los buques, diques, estructuras de alta mar, la ayuda de la toma de decisiones en la gestión costera, entre otros.

Otro aspecto práctico se enfoca en el área educativa, videojuegos, simulación oceánica en realidad aumentada, entre otras herramientas. También son utilizados para los efectos visuales de las películas y series. Uno de estos

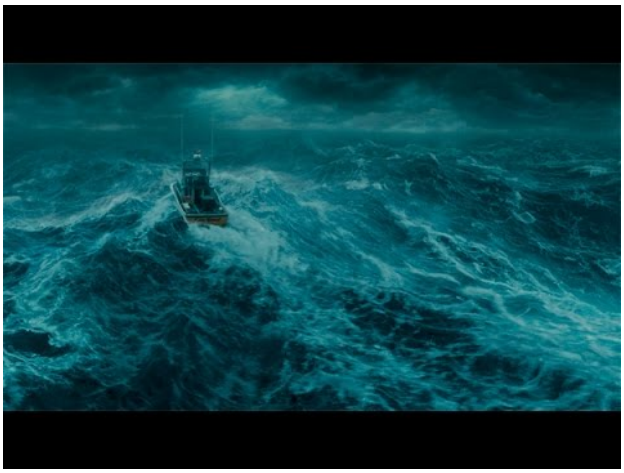

Figura 1: La tormenta perfecta dirigida por Wolfgang Petersen en el año 2000

ejemplares está la tormenta perfecta dirigida por Wolfgang Petersen en el año 2000, ilustrada en la figura 1.

OBJETIVO GENERAL

Construir una escena 3D, que contemple y parametrice el comportamiento de las Olas marinas usando los métodos de transformada de Fourier DFT y FFT.

Objetivos específicos

- Implementar el algoritmo de transformada de Fourier Discreta.
- Implementar el algoritmo de transformada Rápida de Fourier.
- Construir una interfaz gráfica para parametrizar los valores de amplitud, longitud, mallado, vector de viento, entre otros.
- Realizar pruebas de rendimiento gráfico durante la renderización.

CAPÍTULO I

OLAS MARINAS

En el siguiente marco teórico se explicará cómo ocurren estos fenómenos y cómo estas son explicadas a través de leyes físicas. Además, se explicará un método aplicado en computación gráfica que simula el comportamiento de las mismas.

I.1 OLA

Una ola es una onda que se presenta en las zonas acuáticas: ríos, mares, oceanos, entre otros; producto de la influencia de una variedad de fuerzas físicas, como por ejemplo: el viento, los sismos, la fuerza gravitacional, la fuerza lunar, entre otros. Estos fenómenos ocurren por la influencia de dos movimientos: la oscilación del medio movido por la onda, por ejemplo,un movimiento circular; y la propagación de la onda a causa de la energía que setransmiten con ella, trasladando la dirección y la velocidad, llamado velocidad de onda.

Un pequeño desplazamiento en la dirección de propagación es ocasionado por el retorno de las partículas en distintas direcciones. La corriente

de los vientos no son las únicas que provocan las olas, sino también corrientes superficiales. Esto es provocado por la fricción con la superficie del agua, ocasionando cierto arrastre, dando lugar a rizaduras en la superficie del agua, esto se conoce como olas u ondas capilares. Cuando la superficie pierde su lisura, el efecto de fricción se intensifica y las pequeñas rizaduras iniciales dejan paso a olas de gravedad. Las fuerzas que tienden a restaurar la forma lisa de la superficie del agua, y que con ella provocan el avance de la deformación son las fuerzas tensión superficial, mientras que la gravedad es la fuerza que tensa y mueve las olas más grandes. La altura de la ola es directamente proporcional a la energía que puede extraer del viento de forma que se produce una realimentación positiva. La altura de las ondas depende de tres parámetros: la velocidad, su persistencia en el tiempo y la estabilidad de su dirección.

Una vez haya ocurrido estos fenómenos ya mencionados, las olas sobre aguas profundas disipan su energía muy lentamente, de forma que alcanzan regiones muy separadas de su formación. Las olas disipan su energía de varias maneras: puede ser por disipación de fricción por corriente superficial, o por disipación a causa de velocidad excesiva del viento provocando una ruptura de las crestas. Por último, la energía termina por disiparse por interacción con la corteza sólida a causa de la poca profundidad del fondo o cuando las olas se estrellan con la costa.

I.1.1 Tipos de Olas

Ondas libres u oscilatorias: son olas en las que el agua no avanza y describe su giro al subir y bajar en el mismo sitio donde se inició el ascenso. Esto se produce en todas las superficie del mar y se deben a las variaciones de nivel de la misma.

Figura 2: Ilustración de una ola marina

Olas forzadas: son olas generadas por las fuertes corrientes de viento y en ocasiones pueden resultar peligrosamente altas como consecuencia de los huracanes en el agua.

Olas de traslación: son olas que se producen cerca de las costa en la zona de agua poca profunda, y que estas al avanzar y colisionar el fondo, se estrellan contra el litoral formando abundante espuma. Cuando regresa al mar se origina la resaca.

Olas sísmicas: son olas producidas por los sismos y estas pueden resultar bastante peligrosas al llegar a la costa. Estos tipos de olas son también denominadas tsunamis. Pueden generarse bajo dos situaciones: una es que el centro de la pertubaci'øn se hundan las aguas, o que éstas se levanten explosivamente.

Estas dos situaciones conllevan a que la ola se produzca una única ola de dimensiones formidables con gran velocidad que puede oscilar en miles de kilómentros por hora y con una altura superior a 20 metros.(1)

I.2 MOVIMIENTO ONDULATORIO

Para poder estudiar con más profundidad la naturaleza de las olas martítimas, es necesario conocer las leyes físicas que las influyen. Una de estas leyes son los movimientos ondulatorios. Una onda básicamente es una perturbación de alguna propiedad medio (densidad, presión, campo electromagnéticom, entre otros). La onda transporta energía. Las ondas se clasifican en función del medio de propagación, la propagación en si, la periodicidad y de su frente de onda.

I.2.1 Las ondas en función de su medio de propagación

Medio material o mecánica: son ondas que se propagan por un medio material. Para que las ondas se propaguen, se necesita de un medio elástico, bien sea líquido, sólido o gaseoso. Las partículas oscilan alrededor de un punto fijo, por lo que no existe transporte neto de materia a través del medio. Por ejemplo, las ondas de gravedad, ondas sonoras y las ondas elásticas.

Medio no mecánico: son ondas que no necesitan de un medio elástico para propgarse, sino que se propagan en el vacío. Las ondas electromagnéticas son ejemplo de ello.

Escalares: son ondas que poseen una magnitud, pero que carece de dirección y sentido. Por ejemplo, la presión de un gas, o la onda emitida por

las partículas elementales de un átomo.

Vectoriales: son ondas que poseen un dirección y sentido. Estas pueden ser: longitudinales, cuyo movimiento de las partículas que transporta las ondas es paralelo a la dirección de propagación de la misma, por ejemplo el sonido; y transversal, cuyas partículas se desplazan perpendicular a la dirección de onda, como por ejemplo las ondas electromagnéticas.

I.2.2 Las ondas en función de su periodicidad

Ondas unidimensionales: son ondas que se propagan en una sola direrección del espacio, como las ondas en los muelles o en las cuerdas. Si se propaga en una dirección única, sus frentes de ondas son planos y paralelos.

Ondas superficiales: Son ondas que pueden propagarse en dos direcciones, es decir en un plano. Por ejemplo, las ondas generadas por el lanzamiento de una piedra en un lago.

Ondas esféricas: son ondas que pueden propagarse en tres direcciones y cuyos frentes de ondas son esferas concéntricas que salen de la fuente de perturbación expandiéndose en todas las direcciones. (2)

I.2.3 Parámetro de los movimientos ondulatorios

Las olas se desplazan de forma ondulatoria como se ilustra en la figura 2. El movimiento ondulatorio tiene una serie de parámetros que las describen tal como se ilustra en la figura 3. Los parámetros son los siguientes:

Amplitud(A): es la distancia que tiene tiene la onda con respecto al punto de equilibrio.

Cresta(C): es el punto más alto de la onda.

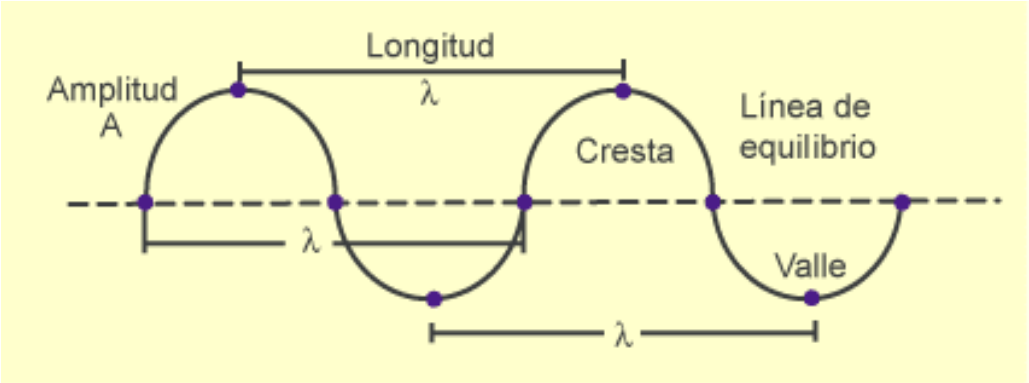

Figura 3: Partes de una onda

Valle(V): es el punto de más bajo de la onda.

Periodo(P): es el tiempo que transcurre en la emisión entre ondas.

Frecuencia(f): es el número de ondas que se emiten en cada instante de tiempo.

Longitud de onda(L): es la distancia entre las crestas y los valles de una onda. (3, 4)

I.3 LA MATERIA EN ESTADO LÍQUIDO

Para entender la causa de estos fenómenos, es bueno estudiar un poco sobre la materia en estado líquido. Cuando las materias se encuentran en estado líquido, las partículas están unidas por fuerza de atracción de menos intensidad que las partículas de estado sólido, de esta manera se mueven con mayor libertad. El número de partículas es muy alto, lo que explica que estos adopten la forma del recipiente por ello las colisiones y las fricciones son bastante notorias. (5)

I.3.1 Colisión entre partículas

Uno de los fenómenos que ocurren en la interacción entre los materiales son las colisiones. Las colisiones son reacciones que ocurren cuando dos o más elementos tangibles interactúan entre sí, de una manera violenta. La colisión entre partículas explica la existencia de los fenómenos como: choques, sumergimiento y por supuesto el oleaje ocasionado por los vientos.

Cuando dos partículas colisionan, generan fuerzas que hacen que el material cambie sus propiedades físicas (y químicas dependiendo del caso). Supongamos la colisión entre el viento (estado gaseoso) y el mar (estado líquido). Cuando se presenta una gran cantidad de masa de aire, las partículas de aire chocan con las partículas de agua, generando fuerzas internas que hace que las partículas se desplacen, lo que trae como consecuencia distintas reacciones mencionados con anterioridad.

Cuando se aplica un soplido al líquido, las partículas del fluido se desplazan en la superficie. Las fuerzas de atracción de las partículas son tan débiles que una simple fricción hace que se desplacen, así como pasa con las olas de viento. Las fricciones causadas por el viento hacen que estas se arrastren.

I.4 TÉCNICAS DE SIMULACIÓN

En la computación gráfica existen diversas técnicas de simulación para ilustrar olas marinas. Una de ellas es la simulación del oleaje usando transformada de fourier(FFT), propuesto en uno de los trabajos de Jerry Tessendorf. Antes de dar introducción a la FFT, es necesario conocer las

ondas de Gertsner, ya que en contraste con la FFT sus matemáticas son relativamente ligeras y claras, y sus conceptos pueden dar una idea aproximada sobre la animación de las olas.

I.4.1 Olas de Gertsner

Las olas de gertsners proveen una solución aproximada a las ecuaciones de fluido. El modelo fíisico tiene como finalidad describir la superficie en términos de movimiento de puntos individuales sobre la superficie. Para una buena aproximación, cada punto posee movimiento circular. Si un punto de una superficie es $x = (x_0, z_0)$ y con una altura $y_0 = 0$, entonces una onda de amplitud A que pasa por el punto sobre la superficie es desplazado en el tiempo t (ver ecuación I.1 y ecuación I.2) .

$$x = x_0 - (\frac{k}{|k|})Asin(k \cdot x_0 - \omega t) \tag{I.1}$$

$$y = Acos(k \cdot x_0 - \omega t) \tag{I.2}$$

Donde se tiene al vector k como el vector de onda, que representa el vector horizontal que apunta en la dirección en que recorre la onda y la magnitud k que se define en la ecuación I.3 con longitud λ.

$$|k| = 2\pi/\lambda \tag{I.3}$$

También incluye el valor de la frecuencia ω. En teoría las ondas de gertsner está limitada a una simple onda senosoidal. Sin embargo, es posible dar un conjunto más complejo a través de la suma de un conjunto de ondas senosoidales. Dando así la siguiente expresión ilustrada en la ecuación I.4.

$$x = x_0 - \sum_{i=1}^{N} (\frac{k}{|k|})A \sin(k \cdot x_0 - \omega_i t + \phi_i) \tag{I.4}$$

I.4.2 Relación de dispersión

El secreto del comportamiento de las olas de gertsner está en la variable de frecuencia ω_i de cada componente. Para el caso de las olas marinas, existe una relación entre la frecuencia de cada componente y la magnitud de sus respectivos vectores de onda, (k_i). Para las aguas profundas, la relación se define como se ilustra en la ecuación 1.5.

$$\omega^2(k) = gk \tag{I.5}$$

Donde g representa la aceleración newtoniana de la gravedad cuya constante es $9.8m/seg^2$. Para las relaciónes de dispersión se tiene muchas condiciones en la que es necesario modificarse. Por ejemplo, para las aguas cuya profundadidad es isómera presenta un retardo que afecta sobre la olas, esto va en contraste con la longitud de onda. Para una relación de dispersión con respecto al nivel de profundidad D, está dada en la ecuación I.6.

$$\omega^2(k) = gk \tanh kD \tag{I.6}$$

Nota que si el nivel de profundidad es muy alto, la relación de dispersión se reduce con respecto a la tangente hiperbólica, determinada en la ecuación I.7.

$$\omega^2(k) = gk(1 + k^2L^2) \tag{I.7}$$

Donde L representa la longitud.Se puede definir una frecuencia base en función de T, como se muestra en la ecuación I.8.

$$\omega_0 \equiv \frac{2\pi}{T} \tag{I.8}$$

Otra de las situaciones es que cuando los vectores de ondas son distribuidos en un mallado regular(en la traducción original regular lattice)

es imposible fijar las frecuencias de dispersión con espacio ω_0. La solución consiste en no usar las frecuencias de dispersión sino en un conjunto que se acerque a estos. Para un vector de frecuencia k, la frecuencia se define como se muestra en la ecuación I.9.

$$\bar{\omega}(k) = \left[\left[\frac{\omega(k)}{\omega_0}\right]\right]\omega_0 \tag{I.9}$$

Donde se define la función [[a]] como la parte entera del valor de a, y la función $\omega(k)$ es una relación de dispersión de interés.

I.4.3 Modelos estadísticos y Transformada de Fourier

Para las olas oceánicas, los modelos estadísticos consideran una función llamada altura de onda como una variable aleatoria de posición horizontal y tiempo. Los modelos estadísticos son usados, en combinación con observaciones experimentales. Basados en la descomposición del campo altura de onda como la suma de senos y cosenos. Computacionalmente, se aplica la Transformada de Fourier(ifft), como método de evaluación rápida. La representación basada en este método se expresa como se ilustra en la ecuación I.10, dado la altura de onda $h(x,t)$ como la variable aleatoria de posición horizontal $x = (x,z)$ y del tiempo t, y se define como la suma de los senos y cosenos(7, p. 3-6).

$$h(x,t) = \sum_{i=0} \bar{h}(k,t) exp(ik \cdot x) \tag{I.10}$$

El parámetro t es el tiempo y k se define como vector bidimensional $k = (kx, kz), kx = 2\pi m/Lx, kz = 2\pi m/Lz$, y n y m son enteros que oscilan entre $-N/2 \leq n < N/2$ y $-M/2 \leq m < M/2$. Por lo cual, dado un punto discreto $x = (nLx/N, mLz/M)$, se define la función campo de amplitud altura de Fourier $\bar{h}(k,t)$), describiendo la estructura de la superficie marina.

El vector pendiente del campo altura es usado para determinar la normal de la superficie, los ángulos de incidencia, y otros aspectos de modelamiento. Una manera de calcularlo es a través de una diferencia finita entre los puntos de mallado de la fft, separado horizontalmente por algún vector bidimensional $\triangle x$. Se define como la pendiente de la superficie acuática, y es determinado como el gradiente del campo altura. Aunque esta diferencia es finita en términos de memoria, puede resultar pobre en cuanto a la aproximación al alcance de ondas con pequeña longitud. Una computación exacta del vector de alcance puede ser obtenido en la ecuación I.11.

$$\epsilon(x,t) = \bigtriangledown h(x,t) = \sum_k ik\bar{h}(k,t)\exp(ik\cdot x). \tag{I.11}$$

En términos de la representación fft, la diferencia finita se reemplazaría el término ik con términos proporcionales a los que se observa en la ecuación I.12.

$$\exp(ik\cdot\triangle x) - 1 \tag{I.12}$$

I.4.4 Espectro de olas

Se define como la distribución de energía en la diferentes frecuencias. Su forma puede estar relacionada con la velocidad del viento, la dirección y la duración (6). Muchos de los análisis estadísticos basados en investigaciones oceano- gráficas se demuestra que las amplitudes de altura $h(k,t)$ están aproximadamente estacionarias, independientes con fluctuaciones gaussianas con un espectro denotado en la ecuación I.13.

$$P(k) = \langle|\tilde{h}^*(k,t)|^2\rangle \tag{I.13}$$

En la que se denota como el promedio encerrado en corchetes $\langle\rangle$.

Existen muchos modelos semi-empíricos para el espectro de ondas $P(k)$. Uno de los modelos útiles está los espectros de Phillips que se ilustra en la ecuación I.14. Este modelo es útil para olas dirigidas por el viento más extensas que las olas capilares en mares completamente grandes.

$$P_h(k) = A\frac{\exp\left(-1/(kL)^2\right)}{k^4}|\hat{k}\cdot\hat{\omega}|^2 \tag{I.14}$$

Donde $L = V^2/g$ es la ola más larga posible que asciende de un viento continuo con velocidad V , mientras que g representa el valor constante de la aceleración de gravedad, $\hat{\omega}$ es la dirección del viento. A es una constante numérica, $|\hat{k}\cdot\hat{\omega}|^2$ es el factor coseno en el espectro de Phillips que elimina el movimiento perpendicular de las olas a la dirección del viento. La desventaja de este modelo es que posee propiedades de convergencia pobre cuando se evalúa un valor alto del número de olas $|k|$. Una solución es suprimir las olas más pequeñas, y modificando el espectro de Phillips por el factor multiplicativo $\exp\left(-k^2l^2\right)$.

I.4.5 Campo de altura de onda oceánica aleatoria

Las amplitudes de fourier de un campo altura pueden ser producto de la ecuación I.15.

$$\tilde{h}_0(k) = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P_h(k)} \tag{I.15}$$

Donde ξ_r y $i\xi_i$ representan números aleatorios gaussianos, con significancia 0 y desviación estándar 1. La distribución gaussiana tiende a seguir datos experimentales en las ondas oceánicas.

Dado una relación de dispersión de ecuación I.5, se define las amplitudes

de los campos de onda realizadas en tiempo t dada en la ecuación 1.16

$$\tilde{h}(k,t) = \tilde{h}_0(k)\exp(i\omega(k)t) + \tilde{h_0^*}(-k)\exp(-i\omega(k)t) \tag{I.16}$$

Esta forma preserva la propiedad de conjugación compleja $\tilde{h}^*(k,t) = \tilde{h}(-k,t)$ al propagar las olas "a la izquierda y a la derecha". Esta forma es también eficiente para computar $h(x,t)$.

I.4.6 Olas picadas

En la representación de la fft, el campo vectorial de desplazamiento 2D permite extrapolar el movimiento horizontal de la ola a la forma 3D. Esto se asocia con la velocidad de la ola y la dinámica de los fluidos. Es computado usando las amplitudes de Fourier de los campos altura. Esta representación es ilustrado en la ecuación I.17.

$$D(x,t) = \sum_k -i\frac{k}{|k|}\tilde{h}(k,t)\exp(ik\cdot x) \tag{I.17}$$

Usando este campo vectorial, la posición de un punto de la superficie es ahora $x + \lambda D(x,t)$, con altura $h(x)$ como antes. Donde λ es el valor de escalamiento para el campo vectorial de desplazamiento.
(7)

I.5 ANTECEDENTES

(a) J. Tessendorf(1999-2004). *Simulating Ocean Water*. Página N^a 7. Este trabajo explica acerca de los modelos estadísticos y las ecuaciones de las Transformada de Fourier(IFFT) para simular escenas de oleaje. La

transformada de Fourier permite evaluar la suma de los senos y cosenos definidos en la ecuaciones de las olas de Gertsner. En los modelos estadísticos se define una variable aleatoria llamada altura de ola, que define el conjunto de campos de altura por cada posición horizontal e instante de tiempo. Para esto se define el campo de amplitud de ola, que depende el vector de ola y del tiempo. También se define la función de desplazamiento que determina el movimiento de la ola. Además, se explica las técnicas de iluminación, radiosidad, reflexión y refracción para escenas marinas, en conjunto con los parámetros y recomendaciones requeridas para el modelado de olas marinas.

(b) C. Yuksel(2010). *Real-time water waves with waves particles*. Página N^a 6. Es un proyecto propuesto en la universidad de Texas, cuyo objetivo es simular comportamientos de agua por medio de una técnica de simulación denominada partículas de olas. Esta técnica captura componentes visuales relevantes a partir de un subconjunto de comportanientos acuáticos. Por lo cual, pueden producirse por medio de la interacción dinámica del agua con cualquier objeto. Esta técnica usa un sistema de partículas para trazar el movimiento de las formas de olas.

(c) J. R. González. *Simulación de oleaje para el estudio de vunerabilidad de obras marítimas*. Página N^a 1. Este se enfoca en el estudio de las obras marítimas, tomando más relevancia el cálculo de la vulnerabilidad de las mismas. Para esto se basa en métodos probabilísticos y pruebas de fallos. El objetivo es crear una metodología que permita simular el oleaje que se puede producir en alta mar a una borrasca o temporal, partiendo de datos incompletos, tomando en cuenta un descriptor global de la misma, caracterizadas por la descripción individual de cada ola. Se describe el oleaje mediante ciertos parametros tales como: la altura de ola, el periodo de oleaje, la máxima sobreevaluación de la ola respecto al

nivel medio, la mínima sobreelevación de la ola respecto al nivel medio, el tiempo transcurrido entre el paso ascendente por cero y el descendente, entre otros parámetros mencionados en este trabajo.

(d) N. P. F. R. F. N. Qizhi Yu(2009).*Feature-Based Vector Simulation of Water Waves*. Página N^a 2. El objetivo de este trabajo consiste en simular olas locales por medio de funciones basadas en simulación vectorial. En este trabajo se habla de animaciones realísticas de cuerpos de agua extensos, que están presentes en ríos, lagos u oceanos, como aspecto importante en las aplicaciones gráficas. Es decir, fenómenos causadas por la colisión entre masas de aguas con cualquier obstáculos. Esto se basan en el algoritmo de Praizelin y Neyret. Además se explica maneras eficientes de mezclar los mallados de las olas y el manejo de la intersección.

CAPÍTULO II

PROPUESTA DE PROYECTO

El proyecto tiene como objetivo construir una escena 3D donde se contemple el comportamiento de las olas marinas, utilizando el método de transformada de fourier. En este capítulo se hará una breve descripción de la propuesta funcional del proyecto de simulación.

En el siguiente diagrama de clase ilustrado en la figura 4, describe la estructura funcional del proyecto de renderización de olas marinas utilizando glew y glfw, en conjunto con las librerías glm para el manejo de matrices. Existe una clase template llamada CBufferAdapter, esta clase es una mejora de la clase CBuffer, capaz de procesar structs en los buffer objects, sin necesidad de guardarlas en arreglos distintos. La implementación se explicará en el siguiente capítulo, por ahora se describirá la estructura funcional del simulador.

El simulador consta de una clase principal llamado cOcean en donde se implementará los algoritmos de la transformada discreta de Fourier y la

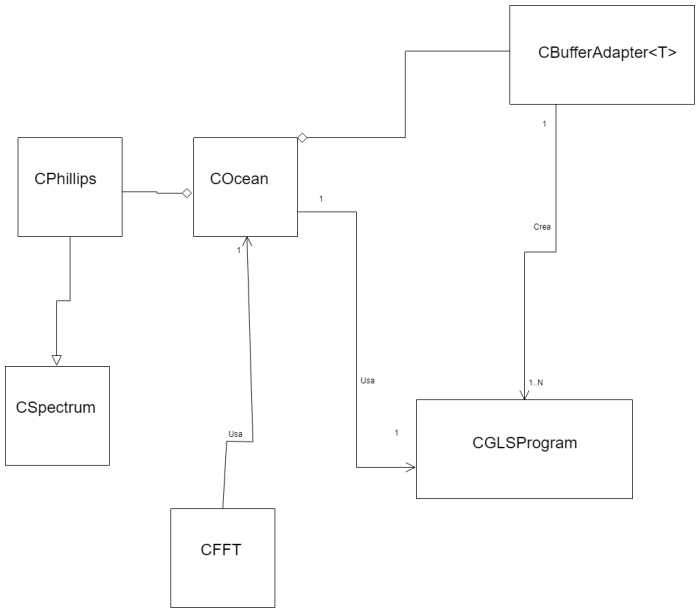

Figura 4: Diagrama de clase de la propuesta del simulador

transformada rápida de Fourier. Involucrando también los métodos que contiene las ecuaciones del espectro de Phillips. A continuación se explica las siguientes clases:

clase cOcean

Permite simular la ola marina. En función del mallado N, amplitud A, vector de viento w.

Métodos:

COcean(N:integer,A:real,w:vec2,length:float,geometry:boolean)

Define una escena oceanica 3D, de dimensión de mallado N(potencia de 2), amplitud A, longitud oceánica *length*.Si se quiere mostrar el mallado se activa o desactiva con geometry.

release()

Limpia toda el espacio de memoria usada para renderizar la escena.

setGLSLProgram(program:CGLSLProgram)

Define el programa shader program uado para renderizar.

dispersion(n_prime:integer,m_prime:integer):float

Calcula la relación de dispersión(ver ecuación I.5) basando en los valores *n_prime* y *m_prime*. Estas son usadas para calcular los vértices *kx* y *kz*.

hTilde_0(n_prime:integer,m_prime:integer)

Calcula el campo de altura de onda oceánica aleatoria(ecuación I.15).

hTilde(t:float, n_prime:integer, m_prime:integer)

Calcula la altura $\tilde{h}(k,t)$ con tiempo t y vértices calculados en n_prime y m_prime(ecuación I.16).

h_D_and_n(x:vec2,t:real):complex_vector_normal

Calcula el valor de desplazamiento *D(x,t)*, el valor de espacio-altura *h(x,t)* y el valor de alcance *c(x,t)*(ecuaciones I.10, I.11,I.17). Esto retorna una estructura de tipo *complex_vector_normal*(ver sección III.1.5).

evaluateWaves(float t)

Calcula los vértices usando DFT en tiempo *t*.

evaluateWavesFFT(t:float)

Calcula los vértices usando FFT en tiempo *t*.

render(t:float,light_pos:vec3,projection:mat4,view:mat4,move:boolean, use_fft:boolean)

renderiza la escena oceanica y simula la ola en tiempo t, aplicando FFT(fft es true) o DFT(fft es false), con proyección projection y vista view.

clase CPhillips

Realiza el cálculo del espectro de Phillips(ecuación I.13).

Métodos:

CPhillips(N:integer,A:float,length:float,w:vec2)

Instancia el modelo del espectro de Phillips con mallado *N*, amplitud *A*, longitud *length* y vector de viento *w*.

$\tilde{C}$Phillips()

Destructor.

createContext(n_prime,m_prime)

Retorna el espectro de Phillips en función de los vértices calculados por *n_prime* y *m_prime*.

clase CFFT

Provee las primitivas para el cálculo de la transformadas rápidas de Fourier(IFFT).

Métodos:

reverse(i:integer)

Cálcula los valores bits reversos, estas son usadas dentro del método fft.

t(x:integer,N:integer)

calcula el factor $T_N^x = \exp\left(\frac{i2\pi x}{N}\right)$. Usado dentro del método fft.

CFFT(N:integer)

Crea una instancia para de transformada de Fourier con matriz de Fourier.

fft(input:array of complex, output:array of complex, stride:integer, offset:integer)

Calcula la transformada rápida de Fourier en función del arreglo input, cuyo resultado es almacenado en *output*, definiendo *offset* desplazamientos partiendo de *x*stride* donde x es una posición índice dada en la iteración interna del algoritmo.

clase BufferAdapter<T>

Es una clase mejorada de CBuffer. Esta clase permite procesar los vértices, normales y coordenadas de textura almacenados en una estructura T(struct vertex_ocean, ver capitulo 3). Sus funciones son similares a las primitivas de CBuffer.

Métodos:

*fillBuffer(nBuffer::bufferId nId, const GLuint nSize, void *npData)*

Rellena npData en la zona nId, cuyo tamaño es nSize bytes de memoria. nId tiene al menos cuatro posibles valores, sin embargo, se menciona los más usados en este proyecto:

(a) VERTEXES_BUFFER: almacena los datos como vértices del modelo.

(b) NORMAL_BUFFER: almacena los datos como las normales del modelo.

(c) TEXTURE_COORDINATES_BUFFER: almacena los datos como coordenadas de texturas para el modelo.

(d) VERTEXES_INDEXES_BUFFER: almacena los datos como índices asociados para el modelo.

draw()

Renderiza el modelo.

drawLine()

Renderiza el mallado del modelo.

drawSubSet(geometry:boolean)

Renderiza los cambios que se efectúan en el modelo bajo el criterio de geometry, es decir, cada cambio que se efectúa en el modelo es procesado como un subconjunto de vértices, normales y coordenadas de textura. Si geometry es true, se renderiza la escena oceánica, si es false, solo se renderiza el mallado.

*fillSubBuffer(void *npData,int size)*

procesa los cambios de modelo con data npData de tamaño size en sus respectivas zonas de memoria de GPU.

clase CGLSProgram

Procesa los programas shaders.

CAPÍTULO III

IMPLEMENTACIÓN

En este capítulo se describe la implementación del simulador, ilustrando las distintas fases de desarrollo como también breves detalles de cada una de las estructuras de datos. No se presentará todo el código del desarrollo, sino que se centrará en las funciones importantes que permite comprender el funcionamiento total de la misma. Antes de comenzar con los detalles, se describe los ajustes que se hacen en las ecuaciones, haciendo más comprensible la fase de desarrollo.

III.1 ANÁLISIS DE LAS ECUACIONES

III.1.1 Análisis del vector k

Se define $k = (2\pi n/Lx, 2\pi m/Lz)$. Cómo $-N/2 \leq n < N/2$ y $-M/2 \leq m < N/2$, entonces se puede escribir como $n = n' - \frac{N}{2}$ y $m = m' - \frac{M}{2}$, de tal manera que $0 \leq n' < N$ y $0 \leq m' < M$, en consecuencia se tendrá vector k como:

$$k = \left(\frac{2\pi(n' - \frac{N}{2})}{L_x}, \frac{2\pi(m' - \frac{M}{2})}{L_z} \right)$$

$$k = \left(\frac{2\pi n' - N}{L_x}, \frac{2\pi m' - M}{L_z} \right) \tag{III.1}$$

III.1.2 Análisis del punto horizontal x

Sea el punto horizontal $x = (x, z)$ cuyo pares ordenados están dados como $x = (nL_x/N, mL_z/M)$. Si $n = n' - \frac{N}{2}$ y $L_x = L_z = N = M$, entonces los puntos horizontales están dadas como lo que se muestra en la ecuación III.2.

$$(x, z) = (n' - N/2, m' - N/2) \tag{III.2}$$

III.1.3 Análisis de la ecuación de altura de onda

Sea $h(x, t)$ el campo de altura de onda ilustrada en la ecuación I.10, si $x = (x, z)$, para todo n' y m' se define el campo altura en la ecuación III.3

$$h(x, z, t) = \sum_{n'=0}^{N-1} \sum_{m'=0}^{M-1} \bar{h}'(n', m', t) \exp\left(\frac{ix(2\pi n' - \pi N)}{L_x} + \frac{iz(2\pi m' - \pi M)}{L_z}\right) \tag{III.3}$$

En la ecuación III.4 define el campo altura cuando N=M=L_x=L_z:

$$h(x, z, t) = \sum_{n'=0}^{N-1} \sum_{m'=0}^{N-1} \bar{h}'(n', m', t) \exp\left(\frac{ix(2\pi n' - \pi N)}{N} + \frac{iz(2\pi m' - \pi N)}{N}\right) \tag{III.4}$$

Para todo n' y m'. Al reorganizar la ecuación aplicando las propiedades de la sumatoria y la exponencial. Se obtiene las ecuaciones III.5.

$$h(x, z, t) = \sum_{m'=0}^{N-1} \exp\left(\frac{iz(2\pi m' - \pi N)}{N}\right) \sum_{n'=0}^{N-1} \bar{h}'(n', m', t) \exp\left(\frac{ix(2\pi n' - \pi N)}{N}\right) \tag{III.5}$$

III.1.4 Amplitudes de los campos de altura

Sea $\tilde{h}(k,t)$ la amplitud de campo de altura para cualquier n' y m', se define la amplitud de campo de altura en las ecuaciones III.6, III.7 y III.8. Donde las ecuaciones I.5, I.10 y I.15 son adaptadas en función de n' y m'.

$$\tilde{h'}(n',m',t) = \tilde{h}'_0(n',m')exp(i\omega'(n',m')t + \tilde{h}'_0(-n',-m')exp(-i\omega'(n',m')t \tag{III.6}$$

$$\omega'(n',m') = \sqrt{g\sqrt{\left(\frac{2\pi n' - \pi N}{L_x}\right)^2 + \left(\frac{2\pi m' - \pi M}{L_z}\right)^2}} \tag{III.7}$$

$$\tilde{h}'_0(n',m') = \frac{1}{\sqrt{2}}(\xi_r + i\xi_i)\sqrt{P'_h(n',m')} \tag{III.8}$$

III.1.5 struct vertex_ocean

Es una estructura de datos en donde se almacena individualmente los valores de los vértices, las normales y la coordenadas de textura del oceano. Esta estructura está descrita de la siguiente manera:

```
struct vertex_ocean{
 GLfloat x, y, z;
 GLfloat nx, ny, nz;
 GLfloat a, b, c;
 GLfloat _a, _b, _c;
 GLfloat ox, oy, oz;
  }
```

Donde los vértices x, y, z son vértices del oceano, mientras que los campos nx, ny, nz almacenas la normales por vértice y los campos a, b, c almacenan las coordenadas de textura.

Los campos adicionales $\_a, \_b, \_c$ y ox, oy, oz representa las coordenadas iniciales de textura y los vértices iniciales del oceano. Estos son usados como variables auxiliares para realizar los cálculos vistos en el capítulo I.

III.1.6 struct complex_vector_normal

Es un tipo de dato estructurado que almacena tres datos importantes:

(a) Campo de altura de onda h(x,t).

(b) el campo vectorial de desplazamiento D(x,t).

(c) vector normal de cada vector x.

La estructura se describe de la siguiente forma:

```
struct complex_vector_normal {
    complex<float> h;
    glm::vec2 D;
            glm::vec3 n;
};
```

Estos campos son llenados al invocar la función h_D_and_n. Esta función solo es aplicada en la transformada discreta de Fourier(DFT).

III.2 ETAPAS DE LA SIMULACIÓN

En la implementación del proyecto, se plantea las siguientes etapas:

III.2.1 Generación de orígenes

Se genera los valores iniciales de los vértices, las normales y las texturas para la onda. Los vértices serán inicializados como vértices de vector k, mientras que las normales tendrán como valor inicial $v = (0, 1, 0)$, mientras que las texturas tendrán como valor inicial la parte real y la parte imaginaria del método *hTilde_0*. Luego, se crea la combinación de índices para generar los mallados. Una vez generado los valores iniciales estas son almacenadas en el GPU usando un buffer, a través de la llamada *fillBuffer*.

III.2.2 Aplicación de las transformadas

Una vez generada los valores iniciales, se aplica la transformada de Fourier. Para este proyecto se implementan los dos métodos: la DFT y la FFT, que pueden ser seleccionados en tiempo de ejecución, por medio de la interfaces que provee *Anttweakbar*.

Las transformadas son aplicadas invocando los métodos *evaluateWaves* y *evaluateWavesFFT* que permite aplicar la transformada discreta y la transformada rápida de Fourier en un tiempo t.

Para la implementación de la DFT, se calcula por cada punto $x = (x, z)$ los valores de campo altura, desplazamiento y normales. Para esto, primero se crea una instancia $x = (x, z)$, luego se aplica las funciones de altura, desplazamiento y normales usando la función *h_D_and_n(x,t)*, luego se asigna los nuevos valores a cada vértice de la escena oceánica con los nuevos valores de vértice de la forma $v = (x + \lambda D(x,t).x, h(x,t).a, z + \lambda D(x,t).z)$.

Para el caso de la FFT, Se determina los vectores $ik\tilde{h}(k,t)$ y los vectores de $-i\frac{k}{k}\tilde{h}(k,t)$. Una vez calculado los vectores, se efectúa el calculo de la transformada rápida de Fourier usando el método fft. Esta clase antes de usarlo, se instancia primero usando el método *cFFT*.

III.2.2.1 Implementación del IFFT

La transformada rápida de Fourier invesa no promediada(IFFT) es un método más eficiente para calcular la transformada discreta de Fourier(DFT), descomponiendo en otras transformadas de Fourier más simple obteniendo dos elementos donde k puede tomar valores entre 0 y 1. (9, 11)

Se puede simplificar la ecuación III.4, aplicando las propiedades de la exponencial se obtiene la ecuación III.9.

$$\begin{aligned} h(x,z,t) &= \sum_{m'=0}^{M-1} \exp\left(\frac{iz(2\pi m' - \pi N)}{N}\right) \sum_{n'=0}^{N-1} \exp\left(\frac{ix(2\pi n' - \pi N)}{N}\right) \bar{h}'(n', m', t) \\ &= \sum_{m'=0}^{N-1} \exp\left(\frac{i2z\pi m'}{N}\right) \exp\left(-iz\pi\right) \sum_{n'=0}^{N-1} \exp\left(\frac{i2x\pi n'}{N}\right) \exp\left(-ix\pi\right) \bar{h}'(n', m', t) \end{aligned} \tag{III.9}$$

Aplicando la definición $\exp(i\pi x) = (\cos\pi + isin\pi)^x = (-1)^x$, se

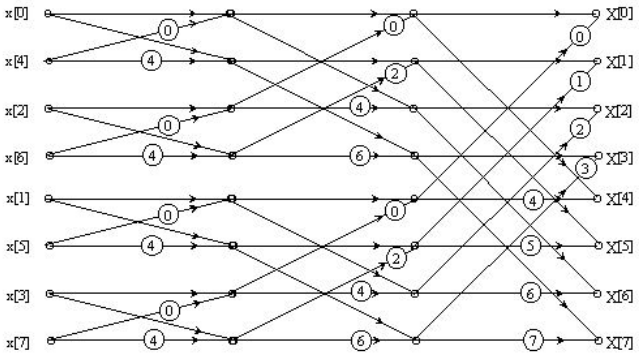

Figura 5: El diagrama Mariposa es una ilustración gráfica de la fft

obtiene la ecuación III.10.

$$\begin{aligned} h(x,z,t) &= \left\{ -1^{z} \sum_{m'=0}^{N-1} \exp\left(\frac{i2z\pi m'}{N}\right) \right\} \left\{ -1^{x} \sum_{n'=0}^{N-1} \exp\left(\frac{i2x\pi n'}{N}\right) \bar{h}'(n',m',t) \right\} \\ &= (-1)^{z} \cdot (-1)^{x} \sum_{m'=0}^{N-1} \exp\left(\frac{i2z\pi m'}{N}\right) \sum_{n'=0}^{N-1} \exp\left(\frac{i2x\pi n'}{N}\right) \bar{h}'(n',m',t) \\ &= (-1)^{x+z} \sum_{m'=0}^{N-1} \exp\left(\frac{i2z\pi m'}{N}\right) \sum_{n'=0}^{N-1} \exp\left(\frac{i2x\pi n'}{N}\right) \bar{h}'(n',m',t) \end{aligned} \tag{III.10}$$

Donde se obtiene el factor de signo $(-1)^{x+z}$. Si se divide las sumatorias de la ecuación III.10 en dos mitades, una transformada de fourier para índices pares y otro para índices impares, se obtiene la ecuación III.11 (10)

$$\begin{aligned} h(x,z,t) = (-1)^{x+z} & \left\{ \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi(2n')x}{N}\right) \tilde{h}'\left(2n',m',t\right) \right. \\ & \left. + \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi(2n'+1)x}{N}\right) \tilde{h}'\left(2n'+1,m',t\right) \right\} \\ & \cdot \left\{ \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{(i2\pi(2m')z}{N}\right) \tilde{h}'\left(n',2m',t\right) \right. \\ & \left. + \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi(2m'+1)z}{N}\right) \tilde{h}'\left(n',2m'+1,t\right) \right\} \end{aligned} \tag{III.11}$$

Si se descompone la sumatoria de indices impares, se halla el coeficiente de la transformada $T_N^x = \exp\left(\frac{i2\pi x}{N}\right)$ como se muestra en las ecuaciones III.12 y III.13.

$$\begin{aligned}
&\sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi(2n'+1)x}{N}\right) \tilde{h'}\left(2n'+1, m', t\right) \\
&= \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi 2n'x}{N}\right) \exp\left(\frac{i2\pi x}{N}\right) \tilde{h'}\left(2n'+1, m', t\right) \\
&= \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi n'x}{\frac{N}{2}}\right) T_N^x \tilde{h'}\left(2n'+1, m', t\right) \qquad \text{(III.12)}
\end{aligned}$$

$$\begin{aligned}
&\sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi(2m'+1)z}{N}\right) \tilde{h'}\left(n', 2m'+1, t\right) \\
&= \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi 2m'z}{N}\right) \exp\left(\frac{i2\pi z}{N}\right) \tilde{h'}\left(n', 2m'+1, t\right) \\
&= \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right) T_N^z \tilde{h'}\left(n', 2m'+1, t\right) \qquad \text{(III.13)}
\end{aligned}$$

Partiendo de las ecuaciones III.12 y III.13 se sustituye en la ecuación III.11. Quedando como resultado la ecuación III.14 . Adicionalmente se simplifica los denominadores de los exponentes aplicando la división, es por ello que existen el denominador $\frac{N}{2}$.

$$\begin{aligned}
h(x,z,t) = (-1)^{x+z} &\left\{ \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi n'x}{\frac{N}{2}}\right) \tilde{h'}\left(2n', m', t\right) \right. \\
&\left. + \sum_{n'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi n'x}{\frac{N}{2}}\right) T_N^x \tilde{h'}\left(2n'+1, m', t\right) \right\} \\
&\cdot \left\{ \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right) \tilde{h'}\left(n', 2m', t\right) \right. \\
&\left. + \sum_{m'=0}^{\frac{N}{2}-1} \exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right) T_N^z \tilde{h'}\left(n', 2m'+1, t\right) \right\} \qquad \text{(III.14)}
\end{aligned}$$

$$h\left(x+\frac{N}{2},z,t\right)=(-1)^{\left(x+\frac{N}{2}\right)+z}\left\{\sum_{n'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi n'\left(x+\frac{N}{2}\right)}{\frac{N}{2}}\right)\tilde{h'}\left(2n',m',t\right)\right.$$
$$\left.+\sum_{n'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi n'\left(x+\frac{N}{2}\right)}{\frac{N}{2}}\right)T_N^{x+\frac{N}{2}}\tilde{h'}\left(2n'+1,m',t\right)\right\}$$
$$\cdot\left\{\sum_{m'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right)\tilde{h'}\left(n',2m',t\right)\right.$$
$$\left.+\sum_{m'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right)T_N^z\tilde{h'}\left(n',2m'+1,t\right)\right\}\quad\text{(III.15)}$$

Si se evalúa la función altura de ola para $x+\frac{N}{2}$, se obtiene el coeficiente $T_N^{x+\frac{N}{2}}$. En la ecuación III.15, se muestra e caso $h\left(x+\frac{N}{2},z,t\right)$. Como $T_N^{x+\frac{N}{2}}=-T_N^x$, entonces la función $h\left(x+\frac{N}{2},z,t\right)$ es definida como la ecuación III.16.

$$h\left(x+\frac{N}{2},z,t\right)=(-1)^{\left(x+\frac{N}{2}\right)+z}\left\{\sum_{n'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi n'\left(x+\frac{N}{2}\right)}{\frac{N}{2}}\right)\tilde{h'}\left(2n',m',t\right)\right.$$
$$\left.-\sum_{n'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi n'\left(x+\frac{N}{2}\right)}{\frac{N}{2}}\right)T_N^x\tilde{h'}\left(2n'+1,m',t\right)\right\}$$
$$\cdot\left\{\sum_{m'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right)\tilde{h'}\left(n',2m',t\right)\right.$$
$$\left.+\sum_{m'=0}^{\frac{N}{2}-1}\exp\left(\frac{i2\pi m'z}{\frac{N}{2}}\right)T_N^z\tilde{h'}\left(n',2m'+1,t\right)\right\}\quad\text{(III.16)}$$

Existen otros casos como por ejemplo: $h\left(x,z+\frac{N}{2},t\right)$ y $h\left(x+\frac{N}{2},z+\frac{N}{2},t\right)$ que se evalúan aplicando la definición mencionada con anterioridad.

Donde En la figura 5, se ilustra el comportamiento de la ola usando FFT para valores N>=2 y para valores N>=4. A medida que se va aumentando el número de mallados, mayor es la subdivisión. Para la primera etapa se descompone en dos DFT de tamaño $\frac{N}{2}$, en la segunda etapas en dos DFT de tamaño $\frac{N}{4}$ y asi sucesivamente. Para eso, se implementa varios métodos

en la clase CFFT, que son claves para su comprensión.(9)

III.2.2.2 Algoritmo del bit reverso

Este algoritmo consiste en reordenar las entradas, de tal manera que el algoritmo de IFFT tenga un resultado deseado. Para esto se aplica los famosos *bit-reverse* que son generados a través del método *reverse(i)*, por medio de operaciones de desplazamiento de bits. En el siguiente código, se visualiza la lógica de como se genera los índices *bit-reverse* (8) necesarios para el algoritmo fft:

```
unsigned int CFFT::reverse(unsigned int i) {
    unsigned int res = 0;
    for (int j = 0; j < log_2_N; j++) {
        res = (res << 1) + (i & 1);
        i >>= 1;
    }

    return res;
}
```

De esta manera, se obtiene la posibilidad reordenar los coeficientes importantes para efectuar la fft.

Calculando los T_N^x

Los cálculos T_N^x son precalculados en el constructor, a través de la función t, a través del siguiente código:

```
  complex<float> CFFT::t(unsigned int x, unsigned int
    N) {
    return complex<float>(cos(pi2*x/N), sin(pi2*x/N));
}
```

En el constructor se implementa el precálculo de los bits inversos y los valores de los T_N^x.

```
CFFT::CFFT(unsigned int N): N(N), reversed(0), T(0),
   pi2(2.00f * M_PI) {
    c[0] = c[1]=0;
```

```
    log_2_N = log(N) / log(2);


    reversed = new unsigned int[N];
       //Calculo de los bit-reverse
    for (int i = 0; i < N; i++) {
        reversed[i] = reverse(i);

    }
    int pow2 = 1;
    //cout << "Matrix W" << endl;
     T = new complex<float> *[log_2_N];  //calcula los
         T
    for (int i = 0; i < log_2_N; i++) {
        T[i] = new complex<float>[pow2];
        for (int j = 0; j < pow2; j++) {
        T[i][j] = t(j, pow2 * 2);

    }
            pow2 *= 2;
        }

                //inicializando el arreglo de
                   coeficientes
          c[0] = new complex<float>[N];
          c[1] = new complex<float>[N];
          for (int i = 0; i < N; i++) {
              c[which][i] = c[which ^ 1][i] = 0;
           }
          which = 0;
}
```

Una vez calculado todos los coeficientes T y los *bit-reversos* se procederá a implementar el método fft:

```
void CFFT::fft(complex<float> *input, complex<float> *
   output, int stride, int offset) {
    fstream fs;
    fs.open("fft.out", ios::app);
    for (int i = 0; i < N; i++) {
        c[which][i] = input[reversed[i] * stride +
           offset];
```

```
    if (isnan(input[reversed[i] * stride + offset
       ].real()) || isnan(input[reversed[i] *
       stride + offset].imag())) {
        cout << input[reversed[i] * stride +
           offset] << endl;
        system("pause");
        exit(0);
    }
    if (isnan(c[which][i].real()) || isnan(c[which
       ][i].imag())) {
        cout << c[which][i] << endl;
        system("pause");
    }
}

int loops       = N >> 1;
int size        = 1 << 1;
int size_over_2 = 1;
int w_          = 0;



for (int i = 1; i <= log_2_N; i++) {
      which ^= 1;
      for (int j = 0; j < loops; j++) {
          for (int k = 0; k < size_over_2; k++) {
              c[which][size*j + k] = c[which ^ 1][
                 size*j + k]
                  + c[which ^ 1][size*j +
                     size_over_2 + k] * T[w_][k];
               //esto es en caso de error
              if (isnan(c[which][size*j + k].real
                 ()) || isnan(c[which][size*j + k].
                 imag())){
                  cout << which << endl;
                  cout << c[which][size*j + k] <<
                     "=" << c[which ^ 1][size*j + k
                     ] << "+" << c[which ^ 1][size*
                     j + size_over_2 + k] << "*" <<
                      T[w_][k] << endl;
                  system("pause");
```

```
                                        exit(1);
                }

            }


        for (int k = size_over_2; k < size; k++) {
            c[which][size*j + k] = c[which ^ 1][size
               *j - size_over_2 + k] -
                c[which ^ 1][size*j + k] * T[w_][k -
                    size_over_2];
              //Esto es en caso de errores
            if (isnan(c[which][size*j + k].real())
               || isnan(c[which][size*j + k].imag()))
               {
                cout << which << endl;
                cout << c[which][size*j + k] << "="
                   << c[which ^ 1][size*j -
                   size_over_2 + k] << "-" << c[which
                    ^ 1][size*j + k]<< "*" <<T[w_][k
                   - size_over_2] << endl;

                   system("pause");
                 exit(1);
            }

           }
        }

        loops       >>= 1;
        size        <<= 1;
        size_over_2 <<= 1;

        w_++;
}


for (int i = 0; i < N; i++) {
    output[i*stride + offset] = c[which][i];
    fs<< c[which][i] << endl;
}
```

}

El método fft consiste en pasar como parámetro de entrada, un arreglo de números complejos perteneciente a una función calculada. Por ejemplo, si se desea obtener los campos de altura $h(x,t)$, es necesario pasar por parámetro de entrada los campos de amplitud de ola $\tilde{h}(k,t)$, para el desplazamiento horizontal los coeficientes $-i\frac{k}{|k|}\tilde{h}(k,t)$. Se inicializa el arreglo c con valores que están en input, cuyas posiciones son los índices calculados por el algoritmo del $bit-reverso$. Una vez inicializado el array, se efectúa por cada cada etapa$_i$ una iteración de j transformadas discretas de fourier, simulando a lo que está ilustrado en la Figura 5 y en la ecuación III.7 .

Una vez aplicado la IFFT estos son retornados en un arreglo de salida *output*, obteniendo arreglo de los campos de altura, los vectores de desplazamiento y las normales. Que son almacenados en los arreglos *h_tilde*, *h_tilde_dx*, *h_tilde_dz*, *h_tilde_slopex* y *h_tilde_slopez*. Por lo cual, actualizarán los vértices y las normales.

```
for (int m_prime = 0; m_prime < N; m_prime++) {
                fft->fft(h_tilde, h_tilde, 1, m_prime*
                   N);
                fft->fft(h_tilde_slopex,
                   h_tilde_slopex, 1, m_prime*N);
                fft->fft(h_tilde_slopez,
                   h_tilde_slopez, 1, m_prime*N);
                fft->fft(h_tilde_dx, h_tilde_dx, 1,
                   m_prime*N);
                fft->fft(h_tilde_dz, h_tilde_dz, 1,
                   m_prime*N);

            }


            for (int n_prime = 0; n_prime < N; n_prime
               ++) {
                fft->fft(h_tilde, h_tilde, N, n_prime)
                   ;
                fft->fft(h_tilde_slopex,
                   h_tilde_slopex, N, n_prime);
                fft->fft(h_tilde_slopez,
                   h_tilde_slopez, N, n_prime);
                fft->fft(h_tilde_dx, h_tilde_dx, N,
```

```
            n_prime);
        fft->fft(h_tilde_dz, h_tilde_dz, N,
            n_prime);

    }
```

Al observar el segmento de código, se aprecia que se aplica FFT en dos iteraciones distintas, esto es similar a una matriz. Primero se aplica N iteraciones con *m_prime* para calcular en columna y N iteraciones con *n_prime*. Esto se hace así para evitar errores de programación en los cálculos y tener legibilidad en la misma.

III.2.3 Renderización

En este último paso, se recolecta los nuevos vértices, las nuevas normales y las nuevas texturas, que son obtenidos a través de las transformadas y luego se actualiza en GPU usando el método *fillSubBuffer*. Una vez actualizada en GPU, se procede con el dibujado de la escena oceánica. Aplicando las matrices correspondientes y adicionalmente generando distintos planos por medio diversas translaciones. El dibujado se aplicado a través del método *drawSubSet*. En el siguiente código se ilustra el renderizado de la escena:

```
void COcean::render(float t, glm::vec3 light_pos, glm
    ::mat4 projection, glm::mat4 View, bool move,bool
    fft) {
//  buffer->fillSubBuffer();


    if (move == true)
        if (!fft)
            evaluateWaves(t);
        else
            evaluateWavesFFT(t);
    //cout << "ciclo infinito" << endl;



    program->use();
    texture->BindTexture();
        Model = glm::mat4(1.0f);
```

```
program->loadUniformMatrix("Projection",
    nGLSLProgram::UNIFORM_SIZE_4X4,glm::
    value_ptr(projection));
program->loadUniformMatrix("View",
    nGLSLProgram::UNIFORM_SIZE_4X4, glm::
    value_ptr(View));
program->loadUniformMatrix("Model",
    nGLSLProgram::UNIFORM_SIZE_4X4, glm::
    value_ptr(Model));
program->loadUniformf("light_position",
    nGLSLProgram::UNIFORM_SIZE_3D, glm::
    value_ptr(light_pos));



//void CBufferAdapter<T>::drawSubSet(nBuffer::
    bufferId nId[], int n, int m, int size)
/*  nBuffer::bufferId nId[3];
nId[0] = nBuffer::VERTEXES_BUFFER;
nId[1] = nBuffer::NORMAL_BUFFER;
nId[2] = nBuffer::TEXTURE_COORDINATES_BUFFER;
    */
double currentTime;
double previousTime;
int frameCounter = 0;
buffer->fillSubBuffer(this->vertices,Nplus1*
    Nplus1);



    for (int j = 0; j < 10; j++)
        for (int i = 0; i < 10; i++) {


            Model = glm::scale(glm::mat4(1.0f),
                glm::vec3(5.0f, 5.0f, 5.0f));
            Model = glm::translate(Model, glm::
                vec3(length * i, 0, length * -j))
                ;
            program->loadUniformMatrix("Model",
                nGLSLProgram::UNIFORM_SIZE_4X4,
                glm::value_ptr(Model));
            buffer->drawSubSet(geometry);
```

Figura 6: Sistema de controles diseñado en Anttweakbar

```
                    /* cout << "valor:" << i*length <<
                       "," << -j*length << endl;
                     system("pause");*/

                    //   system("pause");
                }
        texture->UnBindTexture();
        program->unUse();
}
```

III.2.4 Implementación de la interfaz de parametrización

Para verificar su eficacia, se implementa una interfaz gráfica que permite parametrizar los valores necesarios para la renderización de las olas, usando la librería *Anttweakbar*. En la figura 6, se ilustra todos los parámetros que se escogieron en la simulación.

Los comandos son:

Amplitud: realiza los cambios de amplitud A.

Wind direction:define la direcci ón del viento.

Longitud:define la longitud L.

Camara x:mueve la cámara en coordenada x.

Camara z:mueve la cámara en coordenada z.

Grid Dimension:Define el valor de N.

geometry:Muestra el mallado de la escena.

Aplicar cambios:Guarda los cambios efectuados en la parametrizaciones.

Transformada: Alterna los métodos DFT o FFT.

Simular ola: Activa la simulación.

Una vez que se realiza los cambios en su valores, estas serán actualizadas con el botón *Guardar cambios*. Una vez realizado los cambios deseados, se procede con el botón *Simular Ola*.

CAPÍTULO IV

PRUEBAS Y ANÁLISIS DE RENDIMIENTO

La realización de las pruebas y el análisis de rendimiento son una tarea muy importante en la ciencia de la computación, especialmente en computación gráfica. En este capítulo se describirá los resultados obtenidos después de la implementación de las fases mencionadas en el capítulo anterior, a través de la comparación de ambos métodos que aunque sean eficaces, son diferentes en la eficiencia. Además se explicará el análisis de la eficiencia de las mismas.

IV.1 PARÁMETROS RECOMENDABLES

En esta prueba,se estudia los siguientes aspectos propuestos por Tessendorf:

IV.1.1 El tamaño del mallado N y M

Se recomienda mallados cuadrados de intervalo de 16 y 128, en valores de potencia de dos. Para este proyecto se usó valores entre 64 y 256, por razones de rendimiento. Aunque, los métodos son eficaces para valores muy grandes, tendría resultados no muy contemplados por motivos de hardware.

Figura 7: Desplazamiento horizontal y variación de altura de ola

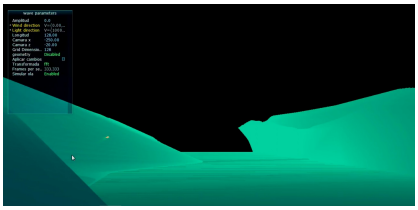

Figura 8: Movimiento en profundidad de la ola oceánica

IV.1.2 Rango de Escala

Para los valores L_x,L_z, M y N deben cumplir ciertas relaciones, se puede asumir que $L_x = L_z$ y que $M = N$. Las direcciones deben ser equivalentes a las relaciones $d_x \equiv L_x/M$ o $d_z \equiv L_z/N$. Por lo general no deben estar por debajo de los 2 cm(0.02) o menos. Además d_z debe ser menor que V^2/g.

IV.2 ANÁLISIS CUALITATIVO

En la figura 7 y 8 se obtiene como resultado la escena oceánica en donde se contemple el comportamiento de las olas marinas. En la figura 7 se puede observar el desplazamiento horizontal y la variación de altura de ola en la escena. Estas variaciones son aleatorias, basados en modelos estadísticos. En la figura 8, se observa que las olas se desplazan en el eje z, esto se debe al definir el vector de ola $k_i = \left(\frac{\pi\left(2n' - \frac{N}{2}\right)}{L}, \frac{\pi\left(2m' - \frac{N}{2}\right)}{L} \right)$.

Teóricamente Tessendorf propone entre 1024 y 2048 para renderizar escenas más realistas. En el caso del titanic, propusieron dimensiones hasta 2048 de mallado. Para poder bajar un poco la intensidad de las fuerzas oceánicas, es recomendable modificar el vector de viento y modificar el valor de la amplitud. Para estas pruebas, no es relevante la definición de estos parámetros.

Mallados(N)	tiempo cpu DFT	tiempo cpu FFT
16	2.0-3.0 seg	0.004-0.005 seg
32	2.9-3.0 seg	0.018-0.020 seg
64	40.0-60.0 seg	0.07-0.008 seg
128	2-3 min	0.35-0.38 seg

Tabla IV.1: Tiempo de consumo de CPU de IDFT y IFFT expresada en segundos(seg)

IV.3 ANÁLISIS CUANTITATIVO

En el análisis cuantitivo se toma lo siguientes aspectos:

IV.3.1 Análisis de rendimiento del método de IDFT y IFFT

La implementación del método *h_D_and_n*, en el cual se calcula los vectores de altura, normales y desplazamiento que solo se calcula la Para el cálculo de estos vectores y valores, se requiere $N * N$ iteraciones, ya que solo la transformada se enfoca las posiciones horizontales (x, z), en vez de las posiciones (x, y, z)(cuya complejidad es O(N^2)), dando cómo resultado una complejidad de $O(N^2)$. Esto trae como consecuencia que para los casos de mallado N=64 es el mejor de los casos, ya que se requiere 4096 para genera los vectores y normales necesarias. Mientras que para el caso N=128, el rendimiento empobrece haciendo que sus resultados de renderizado sean notorios. El método DFT es en el peor de los casos menos eficiente.

Para el análisis de la transformada IFFT, se observa que por cada etapa $k \in [1, log_2(N)]$ se realiza $2^{\frac{N}{2^k}}$ cálculos de dos DFT de $2^j/2$ iteraciones. Por lo cual, se puede deducir que la complejidad es de orden $O(N^2 * log_2(N))$, haciendo una forma más eficiente que el algoritmo DFT. Al observar en la tabla IV.1 existe una gran diferencia entre métodos en cuanto el uso del CPU. Para el método DFT se contempla que a medida que aumente el mallado, el tiempo de procesamiento aumenta; en comparación con el método FFT, el tiempo de procesamiento es ínfimo. Se aspira que esto sea así para los futuros casos. Aunque, existe casos en que el tiempo de procesamiento es constante . Por lo cual, trae una excelente ventaja para el cálculo y el renderizado, sin embargo, si un algoritmo o una estructura de datos no es adecuada para almacenar los cambios de sus vértices, puede desaprovechar

esta ventaja.

CONCLUSIÓN

La simulación de olas marinas es un tema bastante complejo, pero muy desafiante y útil en diversas disciplinas. Estas pueden ser simulados por diversas técnicas entre los cuales se encuentra laI DFT y IFFT. LaI DFT es un método sencillo pero menos eficiente para mallados más grandes, mientras que la FFT es el más efectivo, pero es complejo de comprender y por lo tanto, difícil de implementar. Los métodos difieren mucho en cuanto el tiempo de cómputo y por su eficiencia. Por lo cual, estos puede ser desaprovechados si no se establece los mecanismos y las estructuras de datos adecuadas para almacenar y actualizar los vértices, normales y texturas. La parametrización adecuada de la Amplitud, el vector de viento, la longitud y el mallado debe ser fijadas bajo ciertos criterios. Por ejemplo, para el caso del mallado y la longitud, se necesita que sus relaciones no estén por debajo a un rango específico, de lo contrario los resultados de una escena oceánica son menos de lo esperado.

En el análisis cualitativo, se puede deducir que el realismo de la escena va en función al tamaño del mallado. Es decir, mientras mayor el valor del mallado mayor es el realismo en la escena. Por lo que es importante tener en cuenta los medios más eficientes para procesarlos.

PROYECTOS FUTUROS

En base a los resultados experimentados en este proyecto, se propone otros futuros que abarca este tema. Por lo cual, lo hace desafiante. Entre estos proyectos se tiene:

(a) *Simulación de olas usando programación paralela*: Es la continuación de este proyecto, lo cual tiene como objetivo resolver el tiempo de consumo de CPU en la actualización de los atributos de la escena.

(b) *Simulación de Tsunamis*: este proyecto tiene como objetivo construir una escena 3D, donde se contemple el fenómeno de los tsunamis en base a los parámetros de escalas telúricas, fuerza de gravedad, entre otras.

(c) *Simulación de choques* : tiene como objetivo construir una escena en donde se contemple los choques de las olas con cuerpos sólidos: rocas, barcos, árboles, entre otras.

(d) *Remolinos y tornados*: Simulan la construcción de escenas en las que se contemplen remolinos y tornados.

Referencias Bibliográficas

(1) Ola(2013). Editado el 11 de enero del 2019, https://es.wikipedia.org/wiki/Ola.

(2) Movimiento Ondulatorio(2018). Editado el 17 de octubre del 2018 ,de https://es.wikiversity.org/wiki/Movimiento_Ondulatorio.

(3) La Onda. Sin fecha, de http://cienciasnaturales-fisica.blogspot.com/2007/03/fsica-ii-el-sonido.html.

(4) Características de las Ondas. Sin fecha, de http://www.angelfire.com/empire/seigfrid/Caracteristicasdelasondas.html

(5) Estados de la Materia y el Plasma. Sin fecha, de http://geno-zepeda.webnode.es/estados-de-la-materia-y-plasma-/.

(6) Wave Water(2019). Sin fecha, de https://www.britannica.com/science/wave-water#ref180112 .

(7) J. TESSENDORF(1999-2004), *Simulating Ocean Water*, Editorial desconocido

(8) P. L. ILLESCA y J. M. RODRÍGUEZ(2005), *Lectura 5 Transformada Rápida de Fourier FFT* , 1^a Edición, Valparaiso:Universidad Técnica Federico Santa María Departamento de Electrónica. 9-10 p.

(9) Transformada rápida de Fourier(2018). Editado el 21 de octubre del 2018, https://es.wikipedia.org/wiki/Transformada_rápida_de_Fourier.

(10) K. LANTZ(2011).Ocean Simulation part one: using Discrete Fourier Transform y Ocean Simulation part two: using Fast Fourier Transform. Publicado el 21 de Octubre de 2011, de https://www.keithlantz.net/2011/10/ocean-simulation-part-one-using-the-discrete-fourier-transform/ y de https://www.keithlantz.net/2011/11/ocean-simulation-part-two-using-the-fast-fourier-transform/.

(11) La Transformada rápida de Fourier(s.f). Sin fecha, de http://www.ehu.eus/Procesadodesenales/tema7/ty3.html .