Simulacija oceanskih valova u stvarnom vremenu

Ljubas, Hrvoje

Master's thesis / Diplomski rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:243432

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-23**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 744

SIMULACIJA OCEANSKIH VALOVA U STVARNOM VREMENU

Hrvoje Ljubas

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 744

SIMULACIJA OCEANSKIH VALOVA U STVARNOM VREMENU

Hrvoje Ljubas

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 744

Pristupnik:	**Hrvoje Ljubas (0036522152)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Goran Delač

Zadatak: **Simulacija oceanskih valova u stvarnom vremenu**

Opis zadatka:

Motivirati i opisati problem prikaza oceanskih valova u području računalne grafike. Odabrati, proučiti i opisati trenutna dostignuća kojim se omogućuje prikaz oceanskih valova u stvarnom vremenu. Programski ostvariti sustav koji omogućava generiranje, animiranje i prikazivanje oceanske površine u stvarnom vremenu. Poseban naglasak potrebno je usmjeriti na generiranje površinske geometrije te pri tome primijeniti skup modela proizašao iz oceanografskih istraživanja. Opisati programsko ostvarenje sustava te demonstrirati njegov rad na pokaznom primjeru.

Rok za predaju rada: 14. veljače 2025.

Sadržaj

Uvod 1

1 Opseg rada i povezana područja 2
1.1 Parametrizirani modeli 2
1.2 Spektralni modeli 2

2 Teorijska pozadina 4
2.1 Linearna valna teorija 6
2.1.1 Relacija disperzije 7
2.1.2 Trodimenzionalni val 7
2.1.3 Suma valova 8
2.2 Specifikacija nasumičnog mora 8
2.2.1 Energetski spektar 9
2.2.2 Nasumični proces 9
2.3 Diskretizacija 11
2.3.1 Diskretizacija domene 11
2.3.2 Diskretizacija energije 13
2.4 Valni spektar 14
2.4.1 Pierson Moskowitz spektar 15
2.4.2 JONSWAP spektar 17
2.4.3 Funkcija smjernog širenja 18
2.4.4 Integralna pretvorba domene 19
2.5 Dvodimenzionalni spektri 22
2.5.1 Phillips Spektar 22

3 Implementacija 25
3.1 Osnovne računalne grafike 26
3.2 Sinteza spektra 28
3.3 Diskretna Fourierova transformacija 30
3.3.1 Hermitski spektar 31
3.3.2 Hermitski valni spektar 32
3.3.3 Kombinirana inverzna Fourierova transformacija 33

3.4 Nagibi i pomaci 35
3.4.1 Pomaci 37
3.4.2 Nagibi nakon pomaka 40
4. Sjenčanje 42
4.2 Mapa okoliša 43
4.2 Raspršenje 47
4.3 Morska pjena 48
4.2 Fresnelov efekt 48
4.4 Popločavanje 50
Zaključak 52
Literatura 53
Sažetak 55
Summary 56

Uvod

Simulacija prirodnih fenomena u računalnoj grafici predstavlja izazovnu i kompleksnu temu koja zahtijeva visoku razinu matematičkog modeliranja i računalne snage. Prirodni elementi, poput planina, drveća i vode, sadrže niz složenih struktura i dinamičkih karakteristika koje je teško reproducirati virtualno. Valovi oceana predmet su brojnih istraživanja, budući da njihova simulacija obuhvaća mnoge aspekte kao što su dinamika fluida, refleksija, refrakcija, kao i detalji poput morske pjene. Ovi elementi zajedno doprinose vjerodostojnosti prikaza, što je od ključne važnosti u različitim industrijama, od filmske produkcije i animacija do videoigara i simulacijskog softvera.

Problem realistične simulacije oceanskih valova može se podijeliti u dva osnovna segmenta: generiranje animirane površine mora i sjenčanje te površine. Animacija morske površine zahtijeva kompleksne algoritme kako bi prikazala autentičnu dinamiku valova, uz održavanje računalne učinkovitosti potrebne za interaktivne aplikacije. Sjenčanje, koje uključuje dodavanje boje i svjetlosnih efekata, obuhvaća postupke poput refleksije i refrakcije, pri čemu se simulira način na koji se svjetlost ponaša prilikom interakcije s površinom vode. Refleksija, odnosno odbijanje svjetla s površine mora, zajedno sa refrakcijom, pri kojoj svjetlost prolazi kroz vodu, ključni su za stvaranje dojma dubine i prozirnosti oceana.

Zahvaljujući istraživanjima u oceanografiji, dostupni su precizni matematički modeli koji opisuju ponašanje morskih valova, a koji se mogu iskoristiti za stvaranje uvjerljive simulacije mora u računalnoj grafici. Ovi modeli koriste razne fizikalne principe i statističke metode za generiranje animacija valova. Ovi modeli pružaju realistične rezultate, no često zahtijevaju optimizaciju i prilagodbu kako bi bili računalno izvedivi za aplikacije u stvarnom vremenu.

Ovaj rad ima za cilj istražiti metode i tehnike simulacije oceanskih valova koje se koriste u računalnoj grafici, s posebnim naglaskom na modele temeljenima na fizici i njihovu implementaciju za postizanje uvjerljive simulacije u stvarnom vremenu. Pokazani su različiti pristupi generiranju valova, od jednostavnijih empirijskih modela do kompleksnijih fizičkih simulacija. Kroz analizu ovih metoda, rad također istražuje izazove poput optimizacije za računala s ograničenim resursima, prilagodbu različitim svjetlosnim uvjetima i postizanje efekata poput morske pjene, čime se unaprjeđuje realizam prikaza i povećava gledateljski doživljaj.

.

1 Opseg rada i povezana područja

Oceanografski istraživači klasificiraju ponašanje površine mora prema dubini vode, što rezultira podjelom na plitke, srednje duboke i duboke vode. Ova je podjela važna jer različiti parametri značajno utječu na izgled i dinamiku njihovih površina. Primjerice, površinu plitkih voda karakteriziraju lomljenje valova u blizini obale i interakcije s dnom, dok su površine dubokih voda pod utjecajem primarno vjetra i gravitacijskih sila.

Suvremeni modeli simulacije oceanskih valova svrstani su u tri glavne skupine – parametrizirane, spektralne i CFD modele (engl. *Computational Fluid Dynamics*). Parametrizirani modeli obično se koriste za brže, manje precizne simulacije, dok su spektralni modeli prikladniji za stvaranje detaljnih vizualnih efekata oceanske površine korištenjem Fourierovih transformacija. CFD modeli, iako najpreciznji jer u potpunosti opisuju ponašanje fluida, izostavljeni su u ovom radu zbog izrazito visoke računske zahtjevnosti i ograničene kontrole koju pružaju korisniku pri simulacijama.

1.1 Parametrizirani modeli

Parametrizirani model temelji se na prostornoj domeni, gdje se površina generira i animira kao suma periodičkih funkcija s različitim fazama. Ovaj proces može se izraziti sljedećom formulom:

$$h(x, z, t) = y + \sum_{i=1}^{N} A_i \cos(l_i x + m_i z + \omega_i t) \quad (1.1)$$

, gdje je y srednja visina površine, N je broj valova, A_i je amplitude i-tog vala, ω_i mu je kružna frekvencija, a $\mathbf{k}_i = (l_i, m_i)$ je valni vektor koji definira smjer kretanja vala.

Ovaj pristup, iako jednostavan, zahtijeva ručno biranje amplituda i valnih duljina za svaki val pojedinačno, što ga često čini nepraktičnim za detaljne simulacije oceana zbog složenosti podešavanja potrebnih parametara.

1.2 Spektralni modeli

Oceanografska istraživanja često koriste valne spektre za simulaciju oceanskih površina. Površina oceana predstavlja se kao linearna kombinacija sinusoidnih valova različitih valnih duljina, frekvencija, faza i smjerova, a valni spektar daje distribuciju energije po različitim frekvencijama i valnim duljinama. Ovaj spektar omogućava

realističniji prikaz oceanske površine simuliranjem prirodnog rasporeda valnih energija.

Tessendorf [9] je u svom radu opisao postupak za izračunavanje vertikalnog uzvišenja površine u trenutku **t** na zadanoj koordinati **x** koristeći inverznu diskretnu Fourierovu transformaciju (IDFT). Ovaj pristup omogućava pretvorbu iz spektralne domene natrag u prostornu domenu, pružajući točnu i detaljnu rekonstrukciju oceanskih valova na temelju njihovih spektralnih komponenti.

Formula prema Tessendorfu koristi IDFT za stvaranje oceanskih valova uzimajući u obzir frekvencije i amplitude iz valnog spektra, što rezultira dinamičnom površinom koja oponaša prirodne valne obrasce s visokom razinom preciznosti.

$$h(\mathbf{x}, t) = \sum_{i=1}^{N} A(\mathbf{k}, t) * e^{i\boldsymbol{k}^{\boldsymbol{T}}\mathbf{x}} \qquad (1.2)$$

, gdje **k** predstavlja odabrane valne vektore, a A(**k**,t) predstavlja amplitudu dohvaćenu iz valnog spektra.

Sljedeća poglavlja detaljnije obrađuju različita svojstva valnih spektara. Za sada je važno razumjeti osnovne karakteristike i prednosti koje pružaju.

Valni spektri u oceanografiji često su statični, što znači da svaki spektar definira određeno stanje oceana u skladu sa zadanim parametrima poput brzine vjetra, dubine i udaljenosti od obale. Za različite uvjete mora, kao što je promjena brzine vjetra, potrebno je ponovno sintetizirati spektar. To omogućava modeliranje stanja oceana prilagođavanjem spektra.

Inverzna diskretna Fourierova transformacija (IDFT) koristi se za generiranje valova u spektralnim modelima, dok algoritam brze Fourierove transformacije (FFT) optimizira proces i omogućava brže računanje jednadžbe (1.2). FFT omogućava korištenje većeg broja valnih vektora, čime se postiže preciznija simulacija u odnosu na parametrizirane modele. Iako primjena FFT-a ograničava izbor valnih vektora na uniformno uzorkovanje unutar određenog raspona, što ograničava fleksibilnost u usporedbi s parametriziranim modelima, spektralni modeli su prepoznati kao nadmoćniji zbog svoje učinkovitosti.

Valni vektor opisuje smjer širenja vala, dok valni broj definira njegovu duljinu. Postoje dvije glavne vrste valnih spektara: jednodimenzionalni i dvodimenzionalni. Jednodimenzionalni spektri distribuiraju energiju prema valnom broju, dok dvodimenzionalni spektri distribuiraju energiju prema valnim vektorima, omogućavajući precizniji prikaz smjera širenja valova. Kada se koriste jednodimenzionalni spektri, smjer energije mora biti dodijeljen različitim valnim vektorima pomoću funkcije smjernog širenja. Energija je obično najviše koncentrirana u smjeru puhanja vjetra, što odgovara stvarnim uvjetima u prirodi.

Fokus ovog rada je na spektralnim modelima za duboke vode, jer omogućavaju precizniju simulaciju dinamike oceana.

2 Teorijska pozadina

Površina je generirana utjecajem raznih sila kao što su oluje, potresi, gravitacija te sile Sunca i Mjeseca, s vjetrom kao najistaknutijom silom od svih. Svaka od tih sila djeluje na različitim pojasevima frekvencijskog spektra. Kombinacija tih frekvencijskih pojasa čini spektar jednog vala.

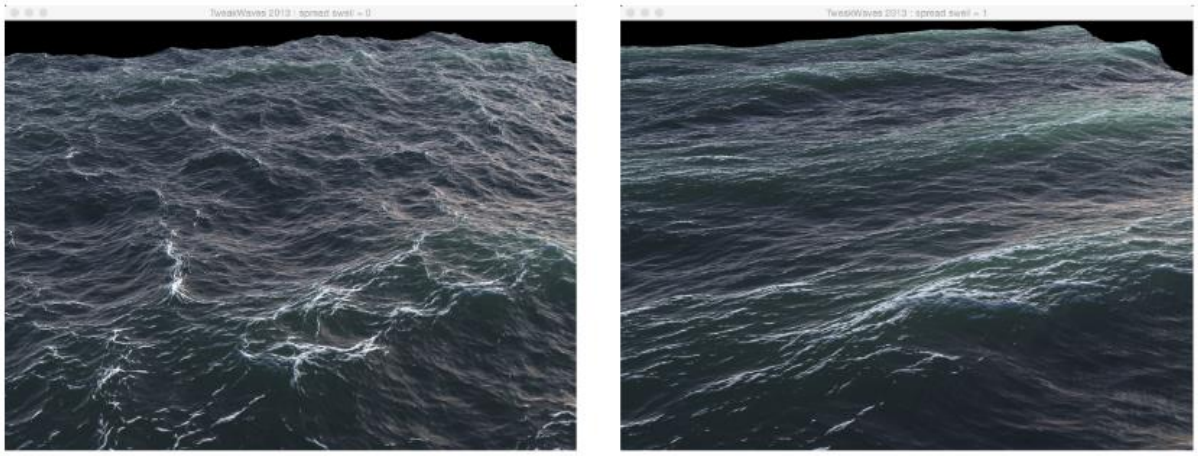
	Period Band Range (s)		Frequency Band Range (Hz)	
Wave Type	Start	End	Start	End
Capillary	0	1×10^{-1}	∞	1×10^{1}
Ultragravity	1×10^{-1}	1×10^{0}	1×10^{1}	1×10^{0}
Gravity	1×10^{0}	3×10^{1}	1×10^{0}	3.33×10^{-2}
Infragravity	3×10^{1}	3×10^{2}	3.33×10^{-2}	3.33×10^{-3}
Long Period	3×10^{2}	4.32×10^{4}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

Tablica 2.1 Sile i njihovi frekvencijski pojasevi djelovanja

Tablica 2.1 daje sažeti pregled različitih vrsta valova na površini oceana, klasificiranih prema njihovim vremenskim periodima i frekvencijama. Najkraći period imaju kapilarni valovi, s trajanjem manjim od 0.1 sekunde. To su prvi valovi koji se opažaju kad vjetar počinje puhati na površini oceana, pojavljujući se kao fino, sitno mreškanje. Slijede ultragravitacijski valovi s periodima od 0.1 do 1 sekunde.

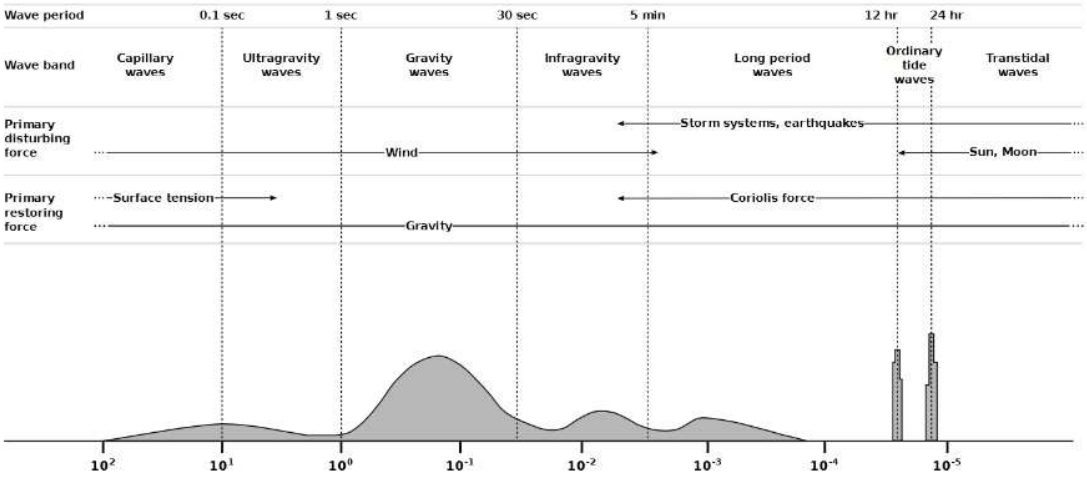
Uobičajeni gravitacijski valovi koji se nalaze na moru imaju periode između 1 i 30 sekundi, a uključuju morske valove i „**uzdužne valove“ (engl. swell)** Morski valovi nastaju pod utjecajem lokalnog vjetra, dok se nakon prestanka vjetra nazivaju uzdužnim valovima. Uzdužni valovi generalno predstavljaju gravitacijske valove koje više ne pokreće lokalni vjetar jer su nastali u drugim područjima ili u nekom trenutku u prošlosti.

Za bolje objašnjenje, slika 2.1 prikazuje razliku između morskih i uzdužnih valova.



Slika 2.1 Ocean dominiran morskim valovima (lijevo), te uzdužnim valovima (desno)

Nakon ovih valova dolaze infragravitacijski valovi, s periodima od 30 sekundi do 5 minuta, dok dugoperiodni valovi obuhvaćaju periode između 5 minuta i 12 sati, među kojima su najpoznatiji tsunamiji i olujni porasti. Astronomske plime predstavljaju obične plimne valove s fiksnim periodima od oko 12 i 24 sata. Na kraju spektra nalaze se transtidalni valovi s periodima dužim od 24 sata, uključujući komponente astronomskih plima s duljim periodima.



Slika 2.2 Približan pregled sila koje utječu na pojedine frekvencijske pojaseve valova, kao i relativne količine energije koju svaki pojas sadrži

Valovi na površini oceana klasificiraju se i prema glavnim silama koje ih pokreću. Prvo, uznemirujuća sila izaziva premještanje vode, dok ju sila vraćanja vraća u početni položaj. Vjetrovi stvaraju kapilarne valove, ultragravitacijske valove, gravitacijske valove i infragravitacijske valove, dok se dugoperiodni valovi poput cunamija

generiraju olujama i potresima. Obične plime uzrokuje gravitacijsko privlačenje Sunca i Mjeseca, dok transtidalne valove pokreću oluje i gravitacijske sile Sunca i Mjeseca.

Nakon što je voda premještena, sila površinske napetosti vraća kapilarne valove u početni položaj, dok gravitacija djeluje kao glavna sila vraćanja za ultragravitacijske, gravitacijske i infragravitacijske valove. Za dugoperiodne valove, obične plimne i transtidalne valove, gravitacija i Coriolisova sila djeluju kao sile vraćanja. Na slici 2.2 prikazan je spektar valova na površini oceana i sile koje ih pokreću. Kao što se vidi, gravitacija je glavna sila vraćanja, pa se valovi na površini oceana nazivaju površinskim gravitacijskim valovima. Većina energije u spektru koncentrirana je u dvama područjima: običnim gravitacijskim valovima i običnim plimnim valovima. Budući da infragravitacijski, dugoperiodni i transtidalni valovi ne doprinose značajno vizualnom izgledu oceana u dubokim vodama, možemo ih izostaviti.

Preostale vrste valova zajednički generira vjetar, zbog čega su poznati pod nazivom vjetrovni valovi. Oni zadržavaju veliku količinu energije u spektru i kreću se od sitnih mreškanja manjim od centimetra do ogromnih valova viših od trideset metara.

Za modeliranje ovako složenog sustava kao što je ocean, potrebno je smanjiti razinu složenosti. Stručnjaci u područjima kao što su oceansko i obalno inženjerstvo koriste Airyjevu teoriju valova [1], poznatu i kao linearna valna teorija. Ova teorija opisuje širenje gravitacijskih valova na površini homogenog fluida pomoću skupa linearnih jednadžbi. Jednadžbe daju dovoljno preciznu aproksimaciju dinamike i kinematike valova kako bi se modeliralo stanje mora u ograničenom vremenskom razdoblju.

2.1 Linearna valna teorija

Linearna teorija valova postavlja nekoliko pretpostavki o svojstvima fluida, poput viskoznosti, stlačivosti i vrtložnosti. Budući da dinamika fluida, povezana svojstva i izvedba linearne teorije valova nadilaze opseg ovog rada, zainteresirani čitatelj može se uputiti na radove Airy-a[1], Batchelor-a[3] i Kinsman-a[2] Međutim, važno je spomenuti dvije ključne pretpostavke linearne teorije valova:

- Tijelo vode ima jednoliku srednju dubinu.
- Amplitude valova su male u odnosu na veličinu tijela vode.

Promatra se idealni ocean u kojem se stalno kreće jedan sinusni val. Parametri koji definiraju sinusni val — amplituda, frekvencija, duljina i smjer — svi su fiksni. Kako bi se pojednostavio matematički opis, pretpostavlja se dvodimenzionalni ocean, gdje jedna dimenzija predstavlja smjer kretanja vala, a druga njegovu vertikalnu promjenu. S ovim pretpostavkama, površinsko izdizanje vode može se opisati sinusoidom:

$$\eta(\mathrm{x},\mathrm{t}) = A * \cos\left(kx - \omega\mathrm{t}\right) \qquad (2.1)$$

Gjde je :

$$k = \frac{2\pi}{\lambda} \qquad \omega = 2\pi * \mathrm{f} \qquad \mathrm{f} = 1 / \mathrm{T}$$

A je amplituda, *k* je valni broj, **x** je horizontalna pozicija, ω je valna frekvencija (rad s^{-1}), t je vrijeme, a λ je valna duljina, f je valna frekvencija (Hz).

Prema linearnoj valnoj teoriji – valni broj *k* i kružna frekvencija **ω** nisu nezavisni parametri, već su povezani relacijom disperzije, koja se obrađuje u nastavku.

2.1.1 Relacija disperzije

U kontekstu valova na površini vode, disperzija se odnosi na disperziju frekvencije, koja opisuje učinak valova različitih valnih duljina koji putuju različitim faznim brzinama.

Definira se funkcijska povezanost ovih dvaju veličina :

$$\omega^2 = g * k * \tanh(k * d) \qquad (2.2)$$

Gdje je ***g*** ubrzane zemljine sile teže, a ***d*** dubina vode. Jednadžba 2.2 naziva se disperzijskom relacijom. Postoje dvije korisne aproksimacije te jednadžbe koje ne uključuju tanh član:

- Aproksimacija za plitku vodu – dubina vode **d** mnogo je manja od valne duljine **λ**. Pretpostavimo d ≪ λ, tada 0 ≤ kd ≪1 i tanh(kd) ≈ kd .
- Aproksimacija za duboku vodu – dubina vode **d** mnogo je veća od valne duljine **λ**. Pretpostavimo d ≫ **λ** , tada kd ≫ 1 i tanh(kd) ≈ 1.

Reducirane disperzijske relacije su sljedeće:

- $\omega^2 = gk^2d$ **Relacije disperzije za plitke vode**
- $\omega^2 = gk$ **Relacija dispezije za duboke vode (2.3)**

2.1.2 Trodimenzionalni val

Do sada se raspravljalo o dvodimenzionalnom oceanu definiranom jednim sinusnim valom. Kao prvi korak prema realističnijem prikazu površine oceana, jednadžba 2.1 proširuje se kako bi oblikovala val u trodimenzionalnom prostoru. Za razliku od jednadžbe 2.1, gdje je smjer kretanja vala fiksiran, ovdje val mora imati mogućnost putovanja u proizvoljnom smjeru na ravnini. Također, potrebno je koristiti dvodimenzionalne ulazne pozicije. Sinusoida koja opisuje izdizanje površine u trodimenzionalnom prostoru definira se na sljedeći način:

$$\eta(\mathbf{x}, \mathrm{t}) = A * \cos(\boldsymbol{k}^{\boldsymbol{T}}\mathbf{x} - \omega \mathrm{t}) \qquad (2.4)$$

Gdje je sada **X** dvodimenzionalna promatrana točka na ravnini i **k** predstavlja putujući smjer vala. Valni broj *k* odgovarajućeg valnog vektora **k** određen je njegovom magnitudom tj. $k = \|\mathbf{k}\|$.

Iz relacije disperzije slijedi da svi valovi sa jednakom magnitudom(valnim brojem) imaju istu kružnu frekvenciju.

Prije nego što se počne baratati sa većim brojem valova, za pojednostavljivanje notacije koristit se Eulerova formula.

$$e^{ix} = \cos(x) + isin(x) \qquad e^{-ix} = \cos(x) - isin(x)$$

gdje se može izvući:

$$\cos(x) = R\{e^{ix}\} \qquad \sin(x) = I\{e^{ix}\}$$

Sada se sinusoida u trodimenzionalnom prostoru može zapisati ovako

$$\eta(\mathbf{x}, \mathrm{t}) = R\{A * \mathrm{e}^{\mathrm{i}[\boldsymbol{k}^{\boldsymbol{T}}\mathbf{x} - \omega \mathrm{t}]}\} \qquad (2.5)$$

Od sada se izostavlja eksplicitno *R* u jednadžbama koje izražavaju η, jer je elevacija površine realan broj.

2.1.3 Suma valova

Nakon što je opisan val u trodimenzionalnom prostoru, potrebno je unaprijediti prikaz površine oceana. Jasno je da ocean nije sastavljen od jednog vala u pokretu, već od beskonačnog broja valova različitih duljina i smjerova. Za vjerodostojan prikaz površine, sasvim je dovoljno koristiti linearnu kombinaciju valova.

Svaka kombinacija amplitude **A**, vektora vala k i frekvencije vala ω definira jedan specifičan val. Sljedećim korakom uklanja se ograničenje jedne amplitude A za sve komponente valova. Konstantna amplituda A zamjenjuje se s a(k), čime se stvara ovisnost između vektora vala i amplitude. Nadalje, zbog relacije disperzije, frekvencija se može izraziti kao funkcija vektora vala, ω=ω(k).

Sada visinu možemo izraziti kao beskonačnu sumu sinusoida.

$$\eta(\mathbf{x}, t) = \iint_k a(k) e^{i(k^T x - \omega(k) t)} dk \qquad (2.6)$$

2.2 Specifikacija nasumičnog mora

Jednadžba 2.6 može opisati površinu oceana, ali uz cijenu definiranja spektra amplituda za svaku točku promatranja na površini mora, što je dugotrajan i neizvediv zadatak. Potrebna je opća formulacija spektra amplituda koja vrijedi za sve točke **x** i vremena t. Srećom, suvremena fizikalna oceanografija temelji se upravo na takvoj formulaciji koju su razvili Neumann i Pierson Jr. [4].

Neumann i Pierson Jr kombiniraju oceanografiju, fiziku i stohastiku kako bi opisali površinu oceana. Temeljna pretpostavka njihove teorije jest da je poremećaj na površini rezultat brojnih doprinosa uzrokovanih relativno nepovezanim silama u različitim trenucima. Stoga je opravdano pretpostaviti da su elementi u zbroju

jednadžbe 2.6 statistički nezavisni. Neumann i Pierson Jr modeliraju površinu oceana pokrenutu vjetrom kao prostorno homogeni i vremenski stacionarni Gaussov slučajni proces [5]. Proces se naziva homogenim jer ne ovisi o odabranoj početnoj točki pozicija $\mathbf{x}$, a stacionarnim budući da apsolutno vrijeme nije relevantno, već je važna samo razlika u vremenu..

2.2.1 Energetski spektar

Ako je dano prostorno homogeno, privremeno stacionarno valno polje i trodimenzionalni spektar amplituda B tada vrijedi.

$$\Theta(\mathbf{k},\omega) = \frac{B(\mathbf{k},\omega)^2}{2}$$

$\Theta(\mathbf{k},\omega)$ predstavlja srednji kvadrat pomaka valne amplitude, koji je direktno povezan s energijom vala jednadžbom:

$$E_w = \rho g * \Theta(\mathbf{k},\omega)$$

Gdje je ρ gustoća vode, a g ubrzanje sile teže. Srednja energija površine se sada može izraziti kao:

$$E_s = \rho g * \iint_k \int_\omega \Theta(\mathbf{k},\omega) d\mathbf{k} d\omega$$

Kako energija vala ovisi o srednjem kvadratu pomaka amplitude vala, tada energija površine ovisi o srednjem kvadratu pomaka površine.

$$\overline{\eta^2} = \iint_k \int_\omega \Theta(\mathbf{k},\omega) d\mathbf{k} d\omega \qquad (2.7)$$

Napomena: Spektar energije, kako je ovdje predstavljen, ograničen je u mogućnosti modeliranja dinamičnog oceana. Dinamičan, u ovom kontekstu, znači ocean koji može započeti kao savršeno ravna površina, biti pokrenut vjetrom do stanja potpuno razvijenog mora, zatim se vratiti u stanje blizu mirovanja i tako dalje. Diskutirani spektar energije modelira samo valove na oceanu beskonačnog prostranstva nad kojim je uniformni vjetar puhao neprekidno. U kontekstu ovog rada, takav model oceana je dovoljan, bez potrebe za simulacijom stvarnog nastanka i raspadanja valova kroz vrijeme.

2.2.2 Nasumični proces

Temeljeno na spektru energije vala, Neumann i Pierson Jr. [4] opisuju površinu kao Gaussov slučajni proces. Kao takav, područje promatranja nije jedna površina η nego čitava skupina površina $\{\eta\}$ koje dijele razdiobu.

Gaussov proces ima asociranu Gaussovu razdiobu sa srednjom vrijednošću $E[\eta]$ i varijancom $Var[\eta]$. Uzimajući u obzir da je površina opisana teorijom linearnih valova s jednakim razmacima između brijega i dola, lako se zaključuje da je $E[\eta] = 0$.

Kako su spektri amplituda statistički neovisni sa srednjom vrijednošću 0 i varijancom $\Theta(\mathbf{k}, \omega)$, varijanca svih komponenti iznosi:

$$Var[\eta] = \iint_k \int_\omega \Theta(\mathbf{k}, \omega) d\mathbf{k} d\omega$$

što je isto kao i jednadžba 2.7. Varijanca σ^2 jednaka je srednjoj kvadratnoj vrijednosti visine površine η^2. Sa poznatom srednjom vrijednošću i varijancom, možemo zapisati Gaussovu distribuciju iz koje su izvedene vrijednosti visine površine mora:

$$pr[\eta] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-\eta^2}{2\sigma^2}}$$

Napomena da se ni položaj ni vrijeme ne pojavljuju u distribuciji, jer je proces modeliran kao prostorno i vremenski nepromjenjiv. Sada se može formulirati visinu površine kao realizaciju stohastičkog procesa.

$$\eta(\mathbf{x}, t) = \iint_k \int_\omega \varepsilon e^{i(\mathbf{k^T x} - \omega t)} d\mathbf{k} d\omega \qquad (2.8)$$

Gdje je ε varijabla Gaussove distribucije sa srednjom vrijednošću E[η] = 0, i varijancom Var[η] = σ^2 definiranom energetskim spektrom Θ(k, ω).

Iz ovoga se može zaključiti da je formuliranje energetskog spektra ključ realističnog prikaza površine.

Iz perspektive algoritamske složenosti, trodimenzionalni energetski spektar je preskup za izračunavanje Fourierovom transformacijom, a jednodimenzionalni spektar ne pruža informacije o smjeru i time sputava u sintezi površine.

Zato se mora koristiti dvodimenzionalni spektar kao bazu. Time se elevacija površine računa kao:

$$\eta(\mathbf{x}, \mathrm{t}) = \iint_k \varepsilon e^{i(\mathbf{k^T x} - \omega(\mathbf{k}) t)} d\mathbf{k} d\omega \qquad (2.9)$$

Jednadžba 2.9 predstavlja dvodimenzionalnu inverznu Fourierovu transformaciju Fourierovih komponenti $\varepsilon e^{-i\omega(\mathbf{k})t}$. Za preostali dio rada, jednadžba 2.9 je osnova za sve daljnje rasprave o modelu spektra valova

2.3 Diskretizacija

Za primjene u računalnoj grafici potrebna je diskretna aproksimacija elevacije površine definirane jednadžbom 2.9, koja daje prihvatljive rezultate te računa te rezultate u razumnom vremenu. Možemo napisati grub nacrt takve diskretne formulacije na sljedeći način:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \varepsilon e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x} - i\omega t)} \qquad (2.10)$$

Jednadžba 2.10 predstavlja inverznu diskretnu Fourierovu transformaciju(DFT) Fourierovih komponenata $\varepsilon * e^{-i\omega t}$, koju možemo izračunati na vrlo učinkovit način pomoću brze Fourierove transformacije (FFT), čija je složenost algoritma o(*nlog(n)*) u odnosu na DFT koji je o(n^2).

Iako FFT algoritam može ublažiti zabrinutost oko performansi, i dalje je potrebno odabrati konačan diskretan skup valnih vektora **k** takav da bude reprezentativan većini energetskog spektra. Također, kako je valni vektor **k** vezan uz valnu duljinu, indirektno je povezan i s prostornom domenom, pa tako i pozicijom **x**. Iz toga slijedi da kvaliteta aproksimacije ovisi o diskretizaciji prostorne domene u konačan skup točaka **x**, kao i domene valnih vektora u konačan set vektora **k**.

2.3.1 Diskretizacija domene

Morska površina definira se kao pravokutno područje, pri čemu širina i visina iznose *L* i *K* metara. Nadalje, diskretizira se spomenuti pravokutnik u rešetku s horizontalnom rezolucijom N i vertikalnom rezolucijom M, pri čemu su i N i M cjelobrojne vrijednosti koje se mogu prikazati kao potencija broja dva. Stoga se definira elevacija morske površine na N × M diskretnih točaka. Razlika između susjednih točaka rešetke horizontalno i vertikalno označuje se kao:

$$\Delta x = \frac{L}{N} \qquad \Delta z = \frac{K}{M}$$

Povišenje se računa na skupu točaka definiranih ovako:

$$\mathbf{x} = (x, z) \in \{(\alpha\Delta\mathrm{x}, \beta\Delta\mathrm{y}) | -\frac{\mathrm{N}}{2} \leq \alpha < \frac{\mathrm{N}}{2}, \frac{-M}{2} < \beta \leq \frac{\mathrm{M}}{2}\} \quad (2.11)$$

Gdje su α i β cijeli brojevi. Za pojednostavljivanje postupka pretpostavlja se da je područje kvadratno tj. da vrijedi: N = M i L = K.

Na temelju *Nyquist-Shannonovog teorema o uzorkovanju* [6], mogu se izračunati minimalne i maksimalne valne duljine koje se mogu rekonstruirati bez gubitka

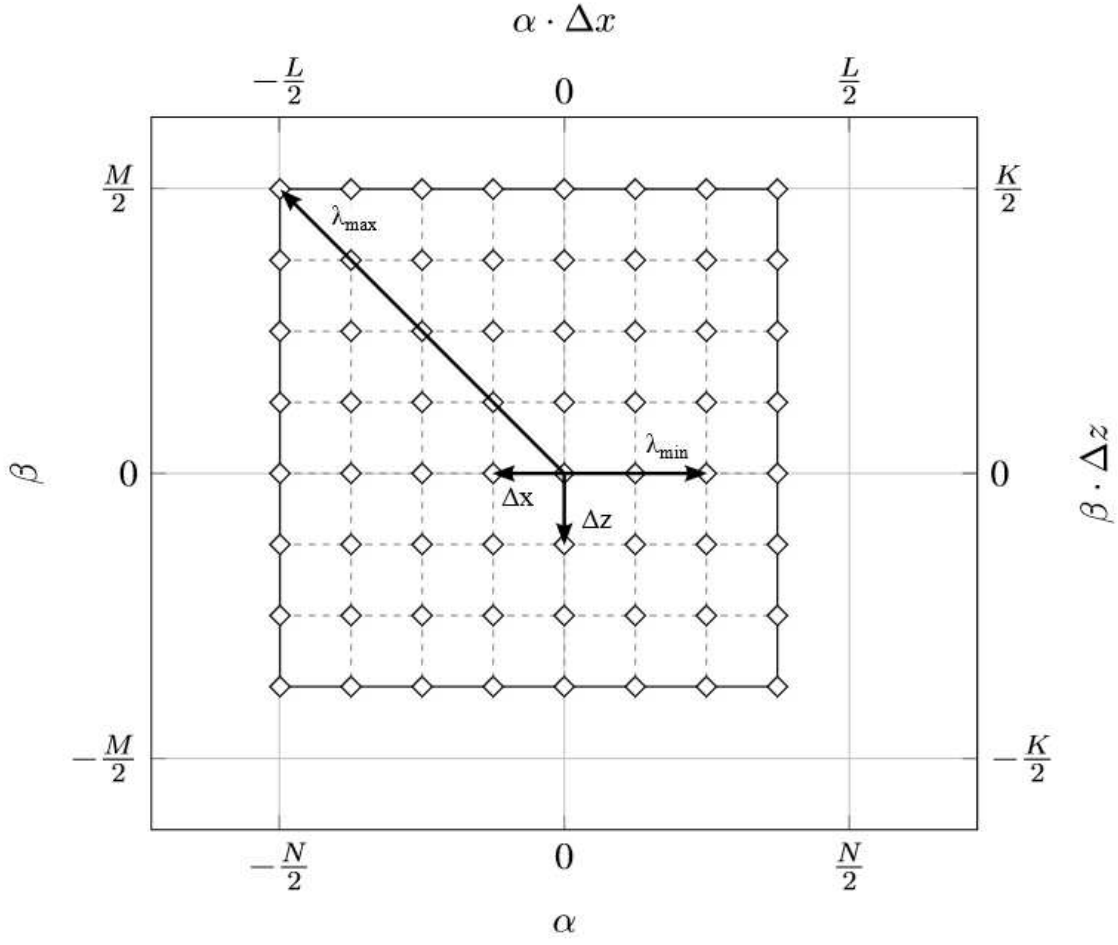
informacija. Navedene minimalne i maksimalne valne duljine mogu se zapisati na sljedeći način:

$$\lambda min = 2\Delta x \qquad \lambda max = \sqrt{2}\,\Delta x\,\frac{N}{2}$$

Te valne duljine imaju i odgovarajuće valne brojeve:

$$k_{min} = \frac{2\pi}{\lambda min}$$

$$k_{max} = \frac{2\pi}{\lambda max}$$



2.3 Primjer diskretizirane prostorne domene

Analogno prostornoj domeni, definiraju se razlike između točaka u domeni valnih vektora :

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

$$\mathbf{k} = (k_x, k_z) \in \{(\alpha\Delta k_x, \beta\Delta k_z) | -\frac{\mathrm{N}}{2} \leq \alpha < \frac{\mathrm{N}}{2}, \frac{-M}{2} < \beta \leq \frac{\mathrm{M}}{2}\} \quad (2.12)$$

Također iz prethodno navedenog vrijedi:

$$k_{min} = \frac{\pi * N}{L} \qquad k_{max} = \sqrt{2}\frac{2\pi}{L}$$

Zaključak je da se odabirom dimenzija L i K implicitno određuje najveću valnu duljinu koja se može obuhvatiti računski, a odabirom N i K određuje se najmanju valnu duljinu koju je moguće prikazati valnim vektorom **k**.

2.3.2 Diskretizacija energije

Područje valnih vektora diskretizirano je u konačan skup, gdje $\pm k_{min}$ predstavlja najveće pozitivne i negativne valne brojeve na horizontalnoj i vertikalnoj osi. Budući da se traži reprezentativnu aproksimaciju ukupne energije vala Var[η] na temelju konačnog skupa valnih vektora, može se započeti ograničavanjem odgovarajućeg integrala s ±kmin.

$$Var[\eta] = \iint_k \Theta(\mathbf{k}) d\mathbf{k} \approx \int_{-k_{min}}^{k_{min}} \int_{-k_{min}}^{k_{min}} \Theta(\mathbf{k}) d\mathbf{k} \qquad (2.13)$$

Sada se ova aproksimacija može napisati kao suma integrala:

$$Var[\eta] = \int_{-k_{min}}^{k_{min}} \int_{-k_{min}}^{k_{min}} \Theta(\mathbf{k}) d\mathbf{k} = \sum_\alpha \sum_\beta \int_{\Delta k_x} \int_{\Delta k_z} \Theta(\mathbf{k}) d\mathbf{k}$$
(2.14)

Gdje su gdje su α i β indeksi iz Jednadžbe 2.12, a Δk_x i Δk_z predstavljaju udaljenosti između točaka mreže. Alternativno, može se izračunati aproksimaciju energije vala na potpuno diskretan način. Neka je $\mathbf{k}_{\alpha,\beta}$ vektor valova definiran jednadžbom 2.12, tada vrijedi

$$Var[\eta] = \sum_\alpha \sum_\beta \Theta(\mathbf{k}_{\alpha,\beta}) \qquad (2.15)$$

Kombiniranjem jednadžbi 2.14 i 2.15 izvodi se

$$\Theta(\mathbf{k}\alpha, \beta) = \int_{\Delta k_x} \int_{\Delta k_z} \Theta(\mathbf{k}) d\mathbf{k} \approx \sum_{\alpha} \sum_{\beta} \Theta(\mathbf{k}\alpha, \beta)\, \Delta k_x \Delta k_x \quad (2.16)$$

Sada se aproksimacija energije može napisati kao jednostavna suma:

$$Var[\eta] = \sum_{\alpha} \sum_{\beta} \Theta(\mathbf{k}\alpha, \beta)\ \Delta k_x \Delta k_x \quad (2.17)$$

Budući da je enrgija vala definirana kao srednji kvadrat amplitude vala, za danu energiju $\Theta(\mathbf{k})$

Moguće je izračunati amplitudu B(k) sljedećim izrazom:

$$B(k)^2 = 2\Theta(\mathbf{k}) \quad (2.18)$$

Kako je opisano u ranijim poglavljima, za generiranje amplitude koristi se Gaussova distribucija temeljena na spektru $\Theta(\mathbf{k})$**.** Budući da sve Gaussove distribucije mogu biti zapisane pomoću normalne distribucije, elevacija površine može se preformulirati u izraz:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \frac{1}{\sqrt{2}} (\varepsilon_r + i\varepsilon_i) \sqrt{2 * \Theta(\mathbf{k}) \Delta k_x \Delta k_z} * e^{i(\mathbf{k^T x} - i\omega t)} \quad (2.19)$$

, gdje su ε_r i ε_i nasumično uzorkovani skalari iz normalne distribucije.

Jednadžbom 2.19 dana je diskretna aproksimacija elevacije površine koja je računski efikasna i daje zadovoljavajuće rezultate za pomno odabrane vektore **k.**

U idućim poglavljima obrađuju se valni spektri, jer oni sačinjavaju osnovu generiranja vizualno uvjerljive morske površine.

2.4 Valni spektar

Oceanografska literatura pruža opširan broj različitih spektara valova za odabir [7]. Za odabir spektra za implementaciju korišteni kriterij je da je spektar dobro utemeljen u oceanografskim istraživanjima i da omogućuje reprodukciju raznovrsnih stanja oceana.

Najčešće nalaženi i citirani spektri u oceanografskoj literaturi su oni jednodimenzionalni. Potrebno ih je pretvoriti u dvodimenzionalne spektre valnih vektora, jer samo ta forma omogućuje stvarnu sintezu ograničenog područja virtualne oceanske površine. Stoga se prvo opisuje kako obogatiti jednodimenzionalne frekvencijske spektre sa smjernim informacijama, a zatim pokazuje kako osigurati

očuvanje integralne jednakosti između različitih domena spektra valova, odnosno kako pretvoriti energiju valova temeljenu na frekvenciji u energiju temeljenu na valnim brojevima s ispravnim skaliranjem.

Također, potrebno se osvrnuti na pojam *potpuno razvijenog mora*.

Razvijeno more je more u kojem je energija koju vjetar prenosi valovima u ravnoteži s prijenosom energije među različitim komponentama valova te s gubitkom energije uslijed lomljenja valova. Svi valovi generirani vjetrom su veliki koliko mogu biti pod trenutnim uvjetima.

Nadalje, razvijeno more neovisno je o „dometu“ vjetra, koji je udaljenost preko koje vjetar puše prije nego dođe do kopna.

S druge strane, modeliranje „mora u razvoju“ mora u obzir uzeti domet vjetra.

U nastavku se obrađuju tri spektra. Od jednodimenzionalnih - Pierson Moskowitz spektar koji modelira potpuno razvijeno more i JONSWAP spektar koji modelira more u razvoju, a od dvodimenzionalnih – Phillips spektar.

2.4.1 Pierson Moskowitz spektar

Pierson-Moskowitzov spektar [10] poznat je i široko korišten jednodimenzionalni frekvencijski spektar temeljen na podacima prikupljenima od vremenskih brodova u sjevernom Atlantiku. U vrijeme kada je uveden, bio je izražen u terminima brzine vjetra na sljedeći način:

$$\Theta(\omega) = \alpha \frac{g^2}{\omega^5} exp[-\beta \left(\frac{g}{U_{19.5} * \omega}\right)^4] \quad (2.20)$$

, gdje je $\alpha = 8.1 * 10^{-3}$ i $\beta = 74 * 10^{-1}$, a $U_{19.5}$ brzina vjetra 19.5 metara iznad morske površine. Daljnja analiza podataka otkrila je sljedeću povezanost između brzine vjetra i vršne frekvencije ω_p :

$$\beta = \frac{5}{4}\omega_p^4$$

što je dovelo do preformuliranja jednadžbe 2.20 u terminima vršne frekvencije:

$$\Theta(\omega) = \alpha \frac{g^2}{\omega^5} exp[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4] \quad (2.21)$$

Gdje $\omega_p = \sqrt[4]{\beta \frac{4}{5}} * \frac{g}{U_{19.5}} \approx 0.877 \frac{g}{U_{19.5}}$

Većina spektralnih modela objavljenih nakon Pierson-Moskowitzovog spektra koristi brzinu vjetra na visini od deset metara iznad morske površine, U_{10}, kao standard. Pretvorba $U_{19.5}$ u U_{10} vrši se na sljedeći način:

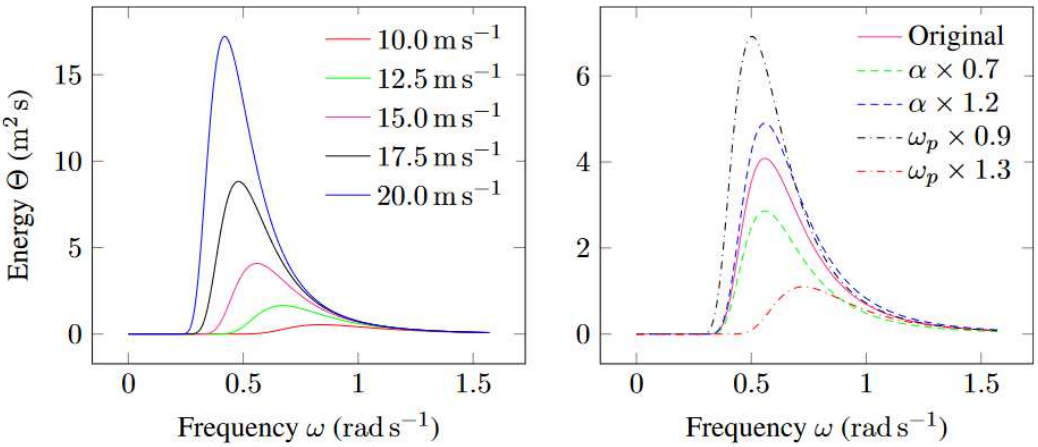
$$U_{10} \approx 1.026 * U_{19.5}$$

Sada je moguće preformulirati vršnu frekvenciju ω u terminima U_{10} :

$$\omega_p \approx 0.855 \frac{g}{U_{10}}$$

Pravilna primjena Pierson-Moskowitzovog spektra ograničena je na potpuno razvijena mora.

Slika 2.4 prikazuje kako se Pierson Moskowitz frekvencijski spektar mijenja ovisno o brzini vjetra, frekvenciji i numeričkoj konstanti α.



Slika 2.4 Promjena Pierson Moskowitz spektra ovisno o parametrima

2.4.2 JONSWAP spektar

JONSWAP spektar je jednodimenzionalni frekvencijski spektar razvijen na temelju podataka prikupljenih tijekom projekta **JO**int **N**orth **S**ea **WA**ve **P**roject. JONSWAP spektar simulira rast valova na temelju putanje vjetra i izražen je kao kombinacija Pierson-Moskowitz spektra i dodatnog faktora pojačanja vrha. Kompaktni oblik JONSWAP spektra često se u literaturi nalazi ovako:

$$\Theta(\omega) = \alpha \frac{g^2}{\omega^5} \, exp\left[-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right] * \gamma^r \qquad (2.22)$$

Gdje je izraz γ^r faktor pojačanja vrha,a vrijedi:

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

$$\alpha = 0.076\left(\frac{gF}{U_{10}^2}\right)^{-0.22}$$

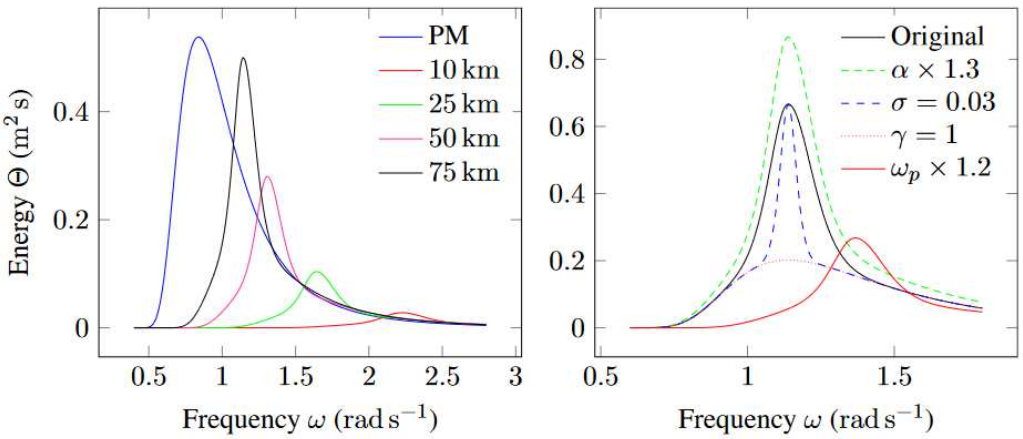
$$\omega_p = 22\left(\frac{U_{10}F}{g^2}\right)^{-0.33}$$

$$\gamma = 3.3$$

$$\sigma = \omega \leq \omega_p \,?\, 0.07 : 0.09$$

Za razliku od Pierson-Moskowitz spektra, koji isključivo modelira potpuno razvijena mora, JONSWAP spektar isključivo modelira mora u razvoju. Hasselman i suradnici [11] navode da u kontekstu njihovog modela more nikada nije potpuno razvijeno, spektar ne konvergira prema Pierson-Moskowitz spektru kako raste dohvat. Slika 2.5 daje okvirni pregled razvoja mora modeliranog JONSWAP-om, uključujući skicu koja prikazuje utjecaj različitih parametara na rezultirajući spektar. Može se primijetiti da spektralni vrh JONSWAP spektra može imati uži oblik od već izraženog vrha Pierson-Moskowitz spektra.

Prije nego što se pređe na dvodimenzionalne spektre, valja pojasniti kako pretvoriti predstavljene jednodimenzionalne frekvencijske spektre u dvodimenzionalne spektre valnog broja.



Slika 2.5 Promjena JONSWAP spektra ovisno o parametrima

2.4.3 Funkcija smjernog širenja

Kao što je napomenuto u ranijim poglavljima, jednodimenzionalni frekvencijski spektar ne sadrži informacije o položaju ili smjeru, što ga čini najjednostavnijim za mjerenje. Međutim, kako bi se moglo sintetizirati konačno područje virtualne površine oceana, potrebne su informacije o smjeru valova. Potrebna je funkcija širenja po smjeru koja energetski spektar Θ(ω) raspodjeljuje po smjerovima te filtrirati valne nizove koji se kreću u suprotnom smjeru od vjetra.

Proizvod običnog jednodimenzionalnog frekvencijskog spektra i funkcije širenja po smjeru može se nazvati frekvencijskim spektrom po smjeru. Frekvencijski spektar po smjeru može se zapisati na sljedeći način :

$$\Theta(\omega,\theta) = \Theta(\omega)D(\omega,\theta) \qquad (2.23)$$

Gdje je :

$$\int_{-\pi}^{\pi} D(\omega,\theta)\,d\theta = 1 \text{ , } \theta \text{ je smjer vala, a } D(\omega,\theta) >= 0 \qquad (2.24)$$

Jednadžba 2.24 osigurava da se količina energije po frekvenciji ne mijenja.

Stoga vrijedi:

$$\Theta(\omega) = \int_{-\pi}^{\pi} \Theta(\omega,\theta)d\theta \qquad (2.25)$$

Literatura pruža veliki skup različitih funkcija smjernog širenja iz kojih se može birati. Odabrana varijanta za analizu je Cosine-2s na temelju rada [11]. Izražena u terminima kutne frekvencije ω, definirana je na sljedeći način:

$$D(\omega,\theta) = \frac{2^{2s-1}}{\pi} \cdot \frac{\Gamma^2(s+1)}{\Gamma(2s+1)} \left|\cos\left(\frac{\theta-\theta_p}{2}\right)\right|^{2s} \qquad (2.26)$$

$$\frac{s}{s_p} = \begin{cases} \frac{\omega}{\omega_p} & za\ \omega \geq \omega_p \\ \frac{\omega}{\omega_p}\ za\ \omega & < \omega_p \end{cases}$$

$$s_p = \begin{cases} 9.77 & za\ \omega \geq \omega_p \\ 6.97\ za & \omega < \omega_p \end{cases}$$

Γ je gamma funkcija, s je faktor širenja, ω_p predstavlja kutnu frekvenciju na kojoj spektar frekvencija ima svoj vrhunac, a θp je smjer valova na tom vrhuncu. Pretpostavlja se da je θp jednak smjeru vjetra.

2.4.4 Integralna pretvorba domene

Prilikom sinteze površine obavlja se prijelaz iz domene valnog broja u prostornu domenu. Stoga je kao preduvjet potreban dvodimenzionalni spektar valnog broja Θ(**k**), no jednadžba 2.23 daje frekvencijski spektar po smjeru Θ(ω,θ). Preoblikuje se jedan oblik spektra u drugi prema teoremu o promjeni varijabli, pri čemu će relacija disperzije za duboke vode služiti kao veza između kutne frekvencije *ω* i valnog broja *k*. Kako bi se izbjegle moguće nejasnoće, dodaju se indeksi svakoj pojavi Θ kako bi se eksplicitno prikazali parametri koje prima kao ulazne vrijednosti. Kao prvi korak, jednodimenzionalni frekvencijski spektar Θω(ω) može se transformirati u jednodimenzionalni spektar valnog broja Θk(k) na sljedeći način:

$$\int \Theta_k(\mathbf{k})\mathrm{d}\mathbf{k} = \int \Theta_\omega(\omega)\mathrm{d}\omega = \int \Theta_\omega\left(\sqrt{gk}\right)\frac{1}{2}\sqrt{\frac{g}{k}}\,\mathrm{dk} \qquad (2.27)$$

Što daje

$$\Theta_k(\mathbf{k}) = \Theta_\omega\left(\sqrt{gk}\right)\frac{1}{2}\sqrt{\frac{g}{k}} \qquad (2.28)$$

S obzirom na Θk(**k**), može se prepisati frekvencijski spektar po smjeru Θω,θ(ω,θ) iz jednadžbe 2.23 u terminima valnog broja k.

$$\Theta(k,\theta) = \Theta(k)D(\sqrt{gk},\theta) \quad (2.29)$$

Kao posljednji korak, treba se prebaciti iz polarnih koordinatna (k,θ) u kartezijanske koordinate valnog vektora **k**. Ta se pretvorba postiže na sljedeći način:

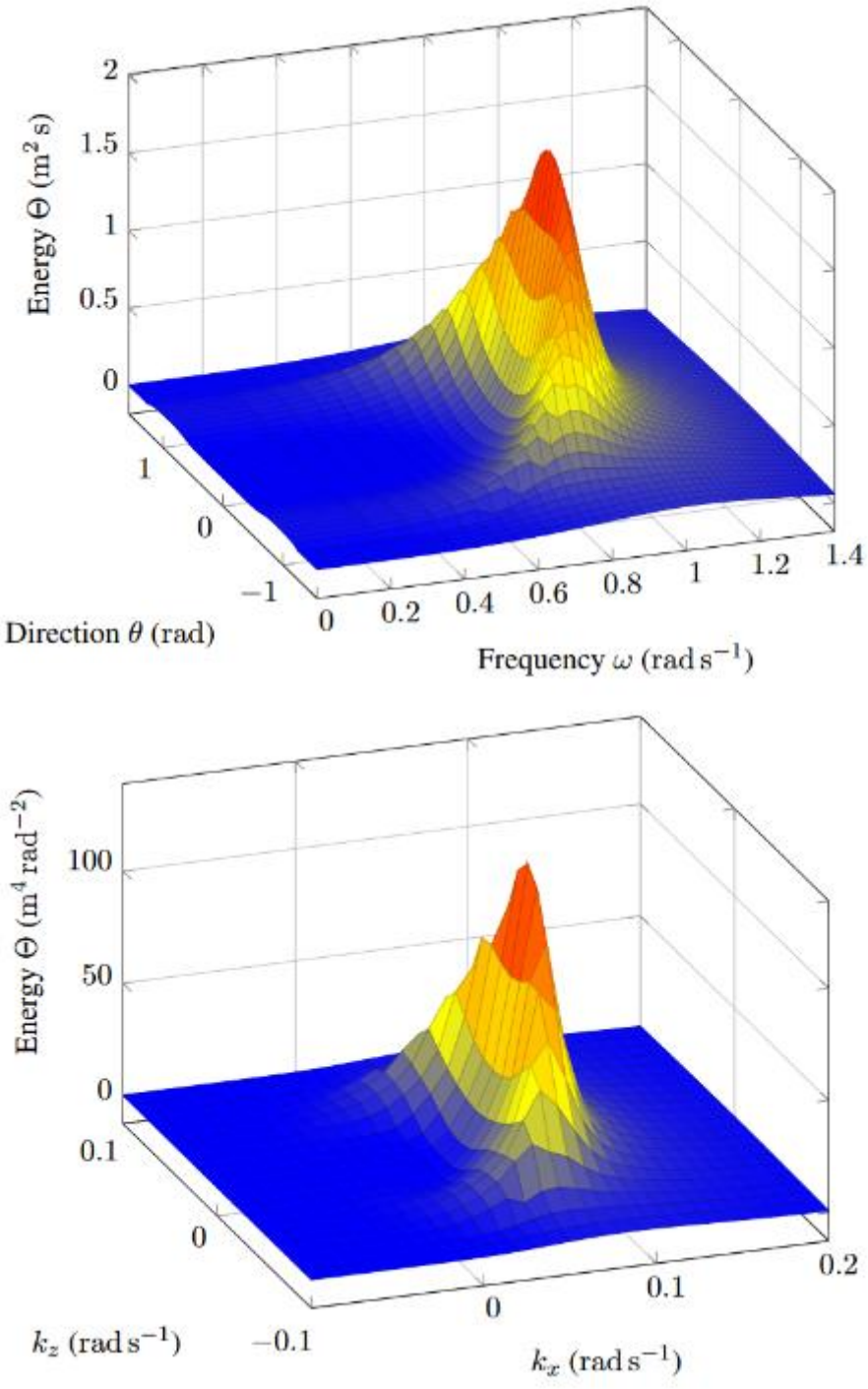
$$\int \Theta_k(\mathbf{k})\mathrm{d}\mathbf{k} = \int\int \Theta_{\mathrm{k},\Theta}(k,\Theta)\frac{1}{k}d\Theta\mathrm{dk} \quad (2.30)$$

Kombiniranje jednadžbi 2.23, 2.28 i 2.30 izvodi se izraz za spektar valnog vektora:

$$\Theta_{\mathbf{k}}(\mathbf{k}) = \Theta_{\omega,\theta}(\sqrt{gk},\theta)\ \frac{1}{2k}\sqrt{\frac{g}{k}}\ ,\ \theta = \arctan\left(\frac{k_z}{k_x}\right) \quad (2.31)$$

Prikaz jednog usmjerenog primjera frekvencijskog spektra te njegovog ekvivalentnog spektra valnih brojeva prikazan je na slici 2.6.

Sada je moguće očuvati integralnu jednakost između različitih prikaza Θ, sve dok se koristi disperzijska relacija za duboke vode.



Slika 2.6 Primjer usmjerenog valnog spektra temeljenog na JONSWAP-u

Primjer frekvencijskog spektra po smjeru kao i njegov ekvivalent u spektru valnog broja po smjeru prikazani su na slici 2.9. Tako je moguće očuvati integralnu jednakost između različitih prikaza Θ pod uvjetom da se koristi relacija disperzije za duboku vodu.

2.5 Dvodimenzionalni spektri

Prethodno je uvedena kombinacija jednodimenzionalnog frekvencijskog spektra s funkcijom širenja po smjeru kako bi se generirao dvodimenzionalni frekvencijski spektar. Pokazano je da postoje različiti oblici dvodimenzionalnih spektra valova, poput frekvencijskog spektra po smjeru Θ(ω,θ), spektra valnog broja u polarnoj formi Θ(*k*,θ) i spektra valnog broja u kartezijskoj formi Θ(**k**). Nadalje, s obzirom na relaciju disperzije, može se prebacivati između svih ovih različitih oblika spektra. Sada se može prijeći na spektre valova koji su eksplicitno uvedeni u dvodimenzionalnoj formi kroz oceanografska istraživanja.

2.5.1 Phillips Spektar

Phillipsov spektar, kako ga je predstavio Tessendorf [1], predstavlja ad-hoc formulaciju dvodimenzionalnog spektra valnog broja Θ(**k**). Razvijen je s ciljem primjene u računalnoj grafici i djelomično se temelji na radovima Phillipsa [12, 13]. Tessendorfova formulacija Phillipsovog spektra može se zapisati na sljedeći način:

$$\Theta(\mathbf{k}) = A\,\frac{\exp\left[-(kL)^{-2} - (kl)^2\right]}{k^4}\,|\mathbf{k}_n^T\mathbf{w}_n|^2 \quad (2.32)$$

Gdje su:

$$k_n = \frac{\mathbf{k}}{k} \qquad\qquad w_n = \frac{\mathbf{w}}{w}$$

„A“ i „*l*“ definirane su kao numeričke konstante, dok **w** predstavlja smjer i brzinu vjetra. Numerička konstanta A označena je kao faktor skaliranja koji je razmotren kasnije. Prema Tessendorfu, L je definiran kao najveći mogući val koji nastaje pod kontinuiranim djelovanjem vjetra brzine $\|\mathbf{w}\|$. Parametrom *l* upravlja se kao frekvencijskim filtrom, čime se kontrolira suzbijanje energije koju pridonose visoki valni brojevi *k*. Da bi se to omogućilo, potrebno je da l bude nekoliko redova veličine manji od L. Sljedeća vrijednost za l odabrana je kao zadana.

$$l = 10^{-3}L$$

Nadalje, iz jednadžbe 2.32 slijedi da postavljanje l = 0 onemogućuje bilo kakvo suzbijanje energije temeljeno na valnom broju. Također, može se primijetiti da Tessendorf koristi jednodimenzionalni spektar valnih brojeva, koji je definiran na sljedeći način:

$$\Theta(k) = A\,\frac{\exp\left[-(kL)^{-2} - (kl)^2\right]}{k^4}$$

Skup instanci jednodimenzionalnog spektra valnog broja prikazan je na slici 2.7. Može se primijetiti da je oblik spektra sličan Pierson-Moskowitzovom spektru na slici 2.5, s time da je frekvencijski raspon vrha spektra smješten na još nižim frekvencijama nego kod Pierson-Moskowitzovog i JONSWAP spektra. Veličina energije, s druge strane, premašuje onu danu Pierson-Moskowitzovim i JONSWAP spektrom za barem faktor 10^3. Podsjeća se da su Pierson-Moskowitzov i JONSWAP model temeljeni na stvarnim podacima dobivenim na oceanu, pa se može zaključiti da ti modeli daju dovoljno precizan raspon vrijednosti energije za stvaranje uvjerljivih rezultata. Stoga je potrebno skalirati energiju dobivenu iz Phillipsovog spektra u taj raspon. Smatralo se razumnim da je to svrha numeričke konstante A.

Budući da Tessendorf ne daje nikakve informacije o A, pokazalo se izazovnim odabrati vrijednost koja skalira energiju prikladno za sve razumne parametre. Na kraju su istraživači odabrali sljedeće prilagodljivo rješenje:

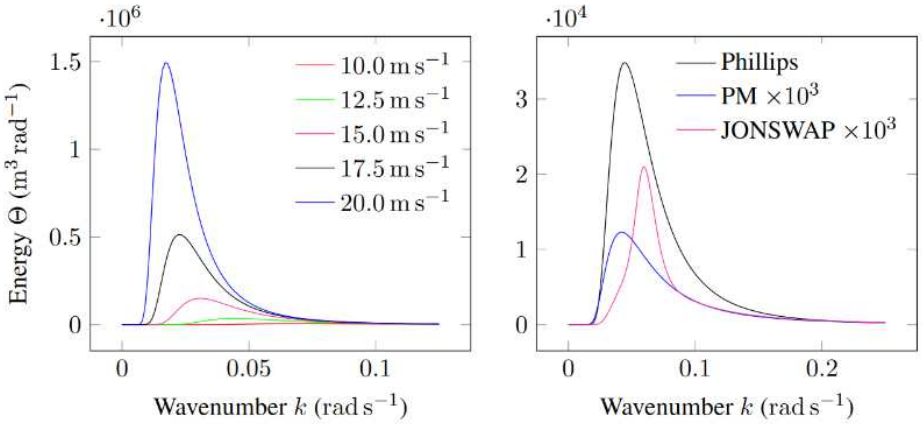
$$A = \frac{0.81}{LK} B \qquad\qquad B \in [0,1]$$

gdje L i K predstavljaju duljine stranica u metrima pravokutnog područja koje se želi sintetizirati. Nakon što je raspravljen jednodimenzionalni spektar valnog broja i faktor skaliranja A, prelazi se na termin za smjer iz jednadžbe 2.32, koji glasi

$$D(\mathbf{k}) = |\mathbf{k}_n^T \mathbf{w}_n|^2 = |\cos\theta|^2$$

ovdje je θ kut između normiranog vektora valova $\mathbf{k}_n$ i normiranog smjera vjetra $\mathbf{w_n}$. Eksponent se može povećati kako bi se rezultirajuće prostiranje u smjeru suzilo. Štoviše, budući da se uzima u obzir samo apsolutna vrijednost kosinusa, rezultat je centro-simetričan. Podsjeća se da prostiranje u smjeru raspodjeljuje energiju predstavljenu s $\Theta(\mathbf{k})$ među svim smjerovima, bez promjene količine energije pri valnom broju **k**. Energija je očuvana, stoga se može zapisati

$$\int_{\pi}^{\pi} D(k,\theta)d\theta = 1 \qquad\qquad D(k,\theta) \geq 0$$



Slika 2.7 Lijevo: Jednodimenzionalni iznos spektra valnog broja Phillipsovog frekvencijskog spektra s različitim vrijednostima brzine vjetra.. Desno: Razlika u skali između Phillipsovog spektra i spektra Pierson-Moskowitz te JONSWAP spektra

3 Implementacija

U prethodnom poglavlju raspravljen je koncept spektra valova, teorijski okvir razvijen kroz oceanografska istraživanja kako bi se opisala raspodjela energije među valovima na površini oceana izloženima istim uvjetima, kao što su brzina vjetra i udaljenost do obale. Na temelju spektra valova može se sintetizirati ocean s velikom raznolikošću površina: od potpuno mirnog mora, preko onoga blago uzburkanog vjetrom, do mora s masivnim valovima izazvanim olujom. No, kako bi se moglo uvjerljivo vizualizirati tako raznolika mora, ovisi se o većem broju podataka od same visine površine. Stoga, kako je opisano u Tessendorfovom radu [1], spektralni prikaz valova može se iskoristiti za dobivanje tri dodatna skupa podataka. Prvo, visokokvalitetne normale dobivene derivacijom vektora nagiba površine oceana u frekvencijskom prostoru. Drugo, pomaci koji transformiraju blagi oblik mora u uzburkaniji, pomoću kojih valovi više nalikuju na stvarne valove nego na sinusoide. Treće, parcijalne derivacije prvog reda prethodno spomenutih pomaka, koji omogućuju određivanje lokacija na površini oceana gdje se može pojaviti pjena. Svaki od tri skupa podataka zahtijeva izradu dodatnog spektra, svih izvedenih iz spektra visine površine oceana.

Pored sinteze podataka, prikazivanje vodene površine velike poput oceana predstavlja izazov. Mogući raspon pogleda kamere može se protezati od krupnih planova površine vode, s detaljima poput mreškanja uzrokovanog gravitacijsko-kapilarnim valovima, do panoramskih pogleda gdje se more proteže sve do horizonta. U prvom slučaju, željena je mogućnost promatranja sitnih detalja na površini vode. Drugi slučaj nije teško riješiti u načelu, budući da su podaci površine oceana koji se sintetiziraju beskonačno ponovljivi. Ipak, poželjno je da gledatelj ne može primijetiti da se površina oceana ponavlja u svim smjerovima čineći pritom efekt popločavanja. Iako se efekt popločavanja ne može potpuno ukloniti kombiniranjem više pločica, barem se može povećati period na najmanji zajednički višekratnik različitih veličina pločica.

Kako se može naslutiti, suočava se sa značajnim brojem Fourierovih transformacija, s obzirom na broj spektara(visina površine, nagibi, pomaci, derivacije prvog reda pomaka) i broj različitih pločica. Stoga se mora osigurati prihvatljiva razina računalnog opterećenja za prikaz u stvarnom vremenu. Problemu se pristupa dvojako. Prvo, ključna svojstva spektra površine oceana omogućuju značajno ubrzanje izračuna njegove inverzne Fourierove transformacije. Drugo, smanjuje se količina spektralnih podataka za transformaciju.

Ostatak ovog poglavlja organiziran je na sljedeći način: U odjeljku 3.1 dan je sažeti pregled osnova računalne grafike potrebnih za razumijevanje tehnika potrebnih za realizaciju implementacije. Odjeljak 3.2 sažeti je pregled sinteze površine oceana putem spektra energije valova. U odjeljku 3.3 opisana su karakteristična svojstva spektra valova koja omogućuju ubrzanje izračuna inverzne Fourierove transformacije. U Odjeljku 3.4 uvedeni su dodatni skupovi podataka temeljeni na spektru valova koji se koriste za prikaz.

3.1 Osnovne računalne grafike

Prije nego što se započne sa stvaranjem grafike, potrebno je stvoriti kontekst i aplikacijski prozor u kojem se crta. Međutim, te su operacije specifične za svaki operacijski sustav. To znači da je ručno potrebno stvoriti prozor, definirati kontekst i upravljati unosom korisnika.

Postoji mnogo biblioteka koje pružaju potrebnu funkcionalnost. Te biblioteke oslobađaju korisnika od rada specifičnog za operacijski sustav i daju prozor i kontekst za renderiranje. Neke od popularnijih biblioteka su GLUT, SDL, SFML i GLFW.

U implementaciji simulacije oceana korišten je GLFW (https://www.glfw.org/).

Primjer inicijalizacije aplikacijskog prozora u C++, pomoću GLFW.

```
1. void Application::initWindow()
2. {
3.     glfwInit();
4.     glfwWindowHint(GLFW_CLIENT_API, GLFW_NO_API);
5.     glfwWindowHint(GLFW_RESIZABLE, GLFW_FALSE);
6.
7.     GLFWmonitor* primaryMonitor = glfwGetPrimaryMonitor();
8.
9.     const GLFWvidmode* mode = glfwGetVideoMode(primaryMonitor);
10.
11.    window = glfwCreateWindow(this->WIDTHo, this->HEIGHTo, "Vulkan Fullscreen",
nullptr, NULL);
12.
13.    glfwSetWindowMonitor(window, primaryMonitor, 0, 0, mode->width, mode->height,
mode->refreshRate);
14.    this->WIDTHo = mode->width;
15.    this->HEIGHTo = mode->height;
16.    glfwSetWindowUserPointer(window, this);
17. }
18.
```

Nadalje, potrebno je uspostaviti komunikaciju sa grafičkom karticom.

Vulkan je API za grafiku nove generacije koji pruža visoko učinkovit pristup modernim GPU-ovima koji se koriste na računalima. (https://www.vulkan.org/) Namijenjen je aplikacijama za 3D grafiku u stvarnom vremenu visokih performansi, poput videoigara i interaktivnih medija, kao i visoko paraleliziranom računarstvu. Pruža značajno niže razine API-ja za aplikaciju u usporedbi sa starijim API-jevima, koja bliže oponaša način na koji moderni GPU-ovi funkcioniraju.

U Vulkan-u se sve odvija u 3D prostoru, ali zaslon ili prozor predstavljaju 2D niz piksela, pa se velik dio rada Vulkan -a odnosi na transformiranje svih 3D koordinata

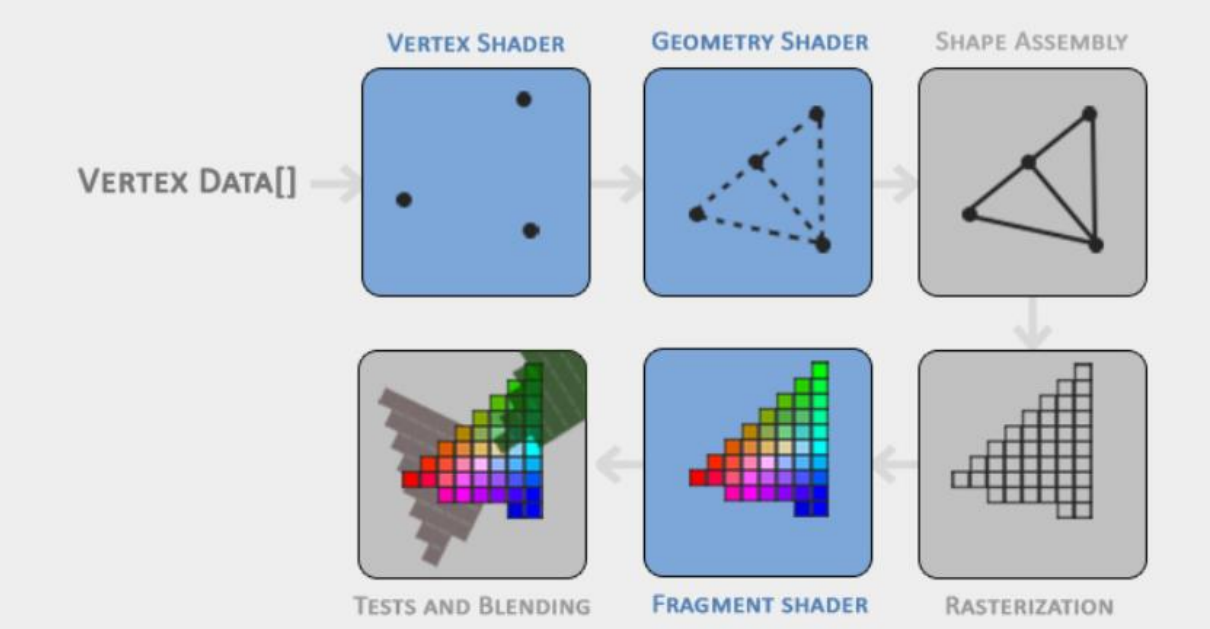
u 2D piksele koji se uklapaju na zaslon. Proces transformacije 3D koordinata u 2D piksele upravlja se grafičkim cjevovodom. Grafički cjevovod može se podijeliti na dva velika dijela: prvi transformira 3D koordinate u 2D koordinate, a drugi dio pretvara te 2D koordinate u stvarne obojene piksele. U ovom poglavlju ukratko se razmatra grafički cjevovod i kako ga se može iskoristiti za generiranje željenih piksela.

Grafički cjevovod kao ulaz prima skup 3D koordinata i transformira ih u obojene 2D piksele na vašem zaslonu. Cjevovod se može podijeliti u nekoliko koraka, pri čemu svaki korak zahtijeva izlaz prethodnog koraka kao svoj ulaz. Svi su ti koraci vrlo specijalizirani (imaju jednu specifičnu funkciju) i lako se mogu izvoditi paralelno. Zbog svoje paralelne prirode, današnje grafičke kartice imaju tisuće malih jezgri za brzo procesiranje podataka unutar grafičkog cjevovoda. Te jezgre pokreću male programe na GPU-u za svaki korak cjevovoda. Ti se mali programi nazivaju sjenčari.

Neki od tih sjenčara mogu se konfigurirati od strane programera, što omogućuje pisanje vlastitih sjenčara za zamjenu postojećih zadanih sjenčara. Pisanje sjenčara ključan je dio uspostavljanja kontrole nad određenim dijelovima cjevovoda.

Sjenčari se pišu u OpenGL Shading jeziku (GLSL).

Ispod je prikazana apstraktna reprezentacija svih faza grafičkog cjevovoda. Napominje se da plavi dijelovi predstavljaju dijelove u koje je moguće umetnuti vlastite sjenčare.



Slika 3.1 Dijelovi grafičkog cjevovoda

Ukratko je objašnjen svaki dio cjevovoda na pojednostavljen način kako bi se pružio dobar pregled načina na koji cjevovod funkcionira.

Kao ulaz u grafički cjevovod na slici 3.1 unosi se popis od tri 3D koordinate koje bi trebale formirati trokut u nizu ovdje nazvanom „vrhovi“ tj. „vertex“ . Vrh je skup podataka za svaku 3D koordinatu. Podaci vrha predstavljeni su pomoću atributa vrha koji mogu sadržavati bilo kakve podatke, ali radi jednostavnosti pretpostavlja se da svaki vrh sadrži samo 3D poziciju i neku vrijednost boje.

Kako bi Vulkan znao kako interpretirati skup koordinata i vrijednosti boja, potrebno je naznačiti vrstu renderiranja koja se želi oblikovati s podacima.

Želi li se da se podaci renderiraju kao skup točaka, skup trokuta ili možda samo kao jedna duga linija, potrebno je deklarirati topologiju tzv. primitiva Vulkan-u tijekom kreiranja cjevovoda. Neki od tih topologija su

- VK_PRIMITIVE_TOPOLOGY_TRIANGLE_LIST,
- VK_PRIMITIVE_TOPOLOGY_POINT_LIST,
- VK_PRIMITIVE_TOPOLOGY_LINE_STRIP

i slično.

Osnovni dijelovi cjevovoda koji zanimaju programera su „Vertex Shader“ i „Fragment Shader“ tj. sjenčar vrhova i sjenčar fragmenata.

Glavna svrha sjenčara vrhova je transformacija vrhova objekta u 3D prostoru, a glavna svrha sjenčara fragmenata je računanje konačne boje piksela.

Uz navedene sjenčare, važan je još i računski sjenčar. Za razliku od starijih API-ja, podrška za računske sjenčare u Vulkanu je obavezna. Svrha računskog sjenčara je računalstvo opće namjene. Tradicionalno domena CPU-a, pomoću računskog sjenčara moguće je računalstvo opće namjene (general purpose computing) izvršavati na grafičkoj kartici (GPUGPU).

Prednosti GPUGPU pristupa su visoka rasina paralelizacije te uklanjanje potrebe da se podaci prebacuju sa CPU na GPU, već mogu biti pospremljeni na GPU i tamo se direktno mogu vršiti svi ostali koraci potrebni za renderiranje.

Računski sjenčar odvojen je od ostatka cjevovoda. To znači da ga se može koristiti kad god se to smatra prikladnim. U implementaciji ovog rada korišten je za računanje FFT-a, pomaka te gradijenata.

3.2 Sinteza spektra

Jednadžbom 2.19 definirana je jednadžba površine na idući način:

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} \frac{1}{\sqrt{2}} (\varepsilon_r + i\varepsilon_i) \sqrt{2 * \Theta(\mathbf{k}) \Delta k_x \Delta k_z} * e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x} - \omega t)}$$

gdje su ε_r i ε_i slučajni skalari izvučeni iz standardne normalne distribucije. I domene valnog vektora **k** i prostorna domena **x** imaju rezoluciju N×N, gdje je N prirodan broj i potencija broja dva. Taj je zahtjev prilagodba algoritmu brze Fourierove

transformacije, koji najbrže radi na takvim rezolucijama [14]. Nadalje, prostorna domena **X** ima raspon L×L, pa je razmak između točaka mreže u domeni valnog vektora sljedeći:

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

Jednadžba 2.19 sadrži sljedeći izraz:

$$h_0(\mathbf{k}) = \frac{1}{\sqrt{2}}(\varepsilon_r + i\varepsilon_i)\sqrt{2 * \Theta(\mathbf{k})\Delta k_x \Delta k_z} \qquad (3.1)$$

h_0 predstavlja nasumično generirani spektar koji se temelji na valnom spektru Θ.

Generira se samo jednom pri pokretanju simulacije za određeni spektar. Time se postiže različit izgled simulacije pri svakom pokretanju. Kako h_0 nema vremensku dimenziju, potrebno ju je ukomponirati.

Iz jednadžbi 2.19 i 3.1 proizlazi:

$$h(\mathbf{k}, t) = h_0(\mathbf{k})\, e^{-i\omega(\boldsymbol{k})t} \qquad (3.2)$$

$h(\mathbf{k}, t)$ mora biti iznova računat pri svakom novom okviru.

Implementacija Phillipsovog spektra u C++-u:

```
static float Phillips(const glm::vec2& k, const glm::vec2& w, float V, float A){
        float L = (V * V) / GRAV_ACCELERATION; // largest possible wave for wind
speed V
        float l = L / 1000.0f;        // supress waves smaller than this

        float kdotw = glm::dot(k, w);
        float k2 = glm::dot(k, k);                          // squared length of wave
vector k
        // k^6 because k must be normalized
        float P_h = A * (expf(-1.0f / (k2 * L * L))) / (k2 * k2 * k2) * (kdotw *
kdotw);
        if (kdotw < 0.0f) {
                // wave is moving against wind direction w
                P_h *= 0.07f;
        }
        return P_h * expf(-k2 * l * l);
}
```

Sinteza spektra h_0, generiranog uzorkovanjem iz Gausseovce distribucije, jedanput pri pokretanju aplikacije.

```
1. for (int m = 0; m <= DISP_MAP_SIZE; ++m) {
2.       k.y = (TWO_PI * (start - m)) / L;
3.
4.       for (int n = 0; n <= DISP_MAP_SIZE; ++n) {
5.                k.x = (TWO_PI * (start - n)) / L;
6.
7.                int index = m * (DISP_MAP_SIZE + 1) + n;
8.                float sqrt_P_h = 0;
9.
10.               if (k.x != 0.0f || k.y != 0.0f)
11.                        sqrt_P_h = sqrtf(Phillips(k, wn, V, A));
12.
13.               app->initial.pixels[2*index] = (float)(sqrt_P_h * gaussian(gen)
                  *ONE_OVER_SQRT_2);
14.               app->initial.pixels[2*index + 1] = (float)(sqrt_P_h * gaussian(gen)
                  * ONE_OVER_SQRT_2);
15.
16.                        // dispersion relation \omega^2(k) = gk
17.               app->frequencies.pixels[index] = sqrtf(GRAV_ACCELERATION *
glm::length(k));
18.               }
19.      }
20.
```

3.3 Diskretna Fourierova transformacija

Visina površinske točke računa se danom inverznom diskretnom Fourierovom transformacijom :

$$\eta(\mathbf{x}, t) = \sum_{\mathbf{k}} h(\boldsymbol{k}, t) * e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x})} \qquad (3.3)$$

Gjde je k :

$$\mathbf{k} = (k_x, k_z) \in \{(\alpha\Delta k_x, \beta\Delta k_z) | -\frac{\mathrm{N}}{2} \leq \alpha < \frac{\mathrm{N}}{2}, \frac{-M}{2} < \beta \leq \frac{\mathrm{M}}{2}\}$$

Valni vektor **k** = (0, 0) daje položaj komponente s nultom frekvencijom, koja kodira prosječnu vrijednost signala u prostornom području. U ovom specifičnom slučaju to bi bila srednja visina površine E[η] = 0. To bi značilo da je komponenta nulte frekvencije locirana pored središta domene valnih vektora, gdje su α = 0 i β = 0. Stvarne implementacije algoritma inverzne Fourierove transformacije očekuju komponentu nulte frekvencije u gornjem lijevom kutu tj. kao prvi element dvodimenzionalnog polja.

Budući da spektar predstavlja periodički signal, ponavlja se u beskonačnost u svim smjerovima. Dakle, neovisno o tome gdje se komponenta nulte frekvencije nalazi u domeni, vrijedi:

$$h(\mathbf{k}(\alpha\Delta k\ ,\beta\Delta k\ ),t) = h\big(\mathbf{k}\big((\alpha + l\mathrm{N})\Delta k\ ,(\beta + mN)\Delta k\ \big),t\big) \qquad l,m\ \in Z$$

Sliak 4.2 prikazuje primjer iskorištavanja periodičnosti spektra za manipulaciju spektra u onog pogodnog za implementaciju.

(a)

$+3.6+0.0i$	$+0.5-0.9i$	$-8.3+0.0i$	$+0.5+0.9i$
$-2.0-1.5i$	$-0.2-0.7i$	$+2.7+2.6i$	$+0.1+0.3i$
$-0.5+0.0i$	$+0.5-0.2i$	$+0$	$+0.5+0.2i$
$-2.0+1.5i$	$+0.1-0.3i$	$+2.7-2.6i$	$-0.2+0.7i$

(b)

Q2	Q1
Q3	Q4

(c)

Q2	Q1	Q2	Q1
Q3	Q4	Q3	Q4
Q2	Q1	Q2	Q1
Q3	Q4	Q3	Q4

(d)

Q4	Q3
Q1	Q2

(e)

$+0$	$+0.5+0.2i$	$-0.5+0.0i$	$+0.5-0.2i$
$+2.7-2.6i$	$-0.2+0.7i$	$-2.0+1.5i$	$+0.1-0.3i$
$-8.3+0.0i$	$+0.5+0.9i$	$+3.6+0.0i$	$+0.5-0.9i$
$+2.7+2.6i$	$+0.1+0.3i$	$-2.0-1.5i$	$-0.2-0.7i$

Slika 3.2

3.3.1 Hermitski spektar

Fourierova transformacija realnog unosa je Hermitska – realni dio rezultirajućeg spektra je parna funkcija, dok je imaginarni dio neparna [6].

Funkcija f(n) je parna ako vrijedi f(−n)=f(n).

S druge strane, funkcija f(n) je neparna ako vrijedi f(−n)=−f(n).

Ako je F(k) Fourierova transformacija funkcije f(n), tada vrijedi:

$$F(-k) = F(k)^* \tag{3.4}$$

Gdje '*' je kompleksno konjugirani operator. To svojstvo se može iskoristiti za dodatne optimizacije.

Prednost jednadžbe 3.4 je što omogućuje optimiziranu funkcionalnost transformacije za realne podatke. Na primjer, za direktnu Fourierovu transformaciju realnih podataka potrebno je izračunati samo polovicu spektra, jer je druga polovica implicitno dana kao kompleksno konjugirani par prve polovine. Slično tome, za inverznu Fourierovu transformaciju dovoljna je samo polovica spektra, dok je druga polovica također implicitno dana. Takve optimizirane transformacije donose prednosti u smislu memorijske efikasnosti, jer se veličina spektra prepolavlja, kao i u smislu računske efikasnosti, budući da se obrađuje samo polovica podataka.

Jasno je da je funkcija η realna funkcija, dakle spektar joj je Hermitski. Ako je F(k) Fourierova transformacija funkcije η, tada se može koristiti dano svojstvo. Periodička priroda spektra omogućava pronalazak kompleksno konjugiranih parova.

3.3.2 Hermitski valni spektar

Spektar *h*(k,t) nije Hermitski iz dva razloga. Prvo, realni dio spektra, ponajprije Θ(k), bi trebao biti parna funkcija, što je jedino slučaj ako je funkcija smjernog širenja centrosimetrična. Drugo, svaki element h_0(k) generiran je umnažanjem sa parom nasumičnih brojeva, tako da je gotovo nemoguće postići svojstvo konjugirane kompleksnosti. Kako bi se iskoristili prednosti koje nudi optimizirana inverzna Fourierova transformacija za realne funkcije, potreban je Hermitski spektar. Stoga se jednadžba 3.3 na sljedeći način:

$$h(\mathbf{k}) = \frac{1}{2} h_0(\mathbf{k})\, e^{-i\omega(\boldsymbol{k})t} + \frac{1}{2} h_0(-\mathbf{k})^*\, e^{i\omega(\boldsymbol{k})t} \tag{3.5}$$

što daje Hermitski spektar, neovisan o generiranim slučajnim brojevima i o tome je li spektar energije valova parna funkcija ili ne

Visina površine η izračunata prema jednadžbi 3.3 daje iste rezultate za jednadžbu 3.2 i jednadžbu 3.5. Ipak, jednadžba 3.2 zahtijeva standardnu inverznu Fourierovu transformaciju s potpunim kompleksnim spektrom izvora i kompleksnim nizom rezultata iste veličine. Realni dio niza rezultata predstavlja visinu površine, dok imaginarni dio ne sadrži relevantne informacije i stoga se odbacuje. S druge strane,

jednadžba 3.5 omogućava primjenu spomenute inverzne Fourierove transformacije optimizirane za Hermitski spektar, čime se brzina računa i korištenje memorije poboljšavaju otprilike dvostruko [15].

3.3.3 Kombinirana inverzna Fourierova transformacija

Za dani Hermitski spektar, moguće je primijeniti specijaliziranu inverznu Fourierovu transformaciju koja zahtijeva samo polovicu stvarnog spektra. Za dva dana Hermitska spektra, moguće ih je kombinirati u jedan spektar i primijeniti standardnu inverznu Fourierovu transformaciju za istovremeno dobivanje oba realna signala . Neka su a i b realni signali s identičnim temeljnim prostornim domenama. Fourierova transformacija označena je sa F, a inverzna Fourierova transformacija sa F^{-1}, tada se može zapisati:

$$c = F^{-1}(F(a) + iF(b)) \tag{3.6}$$

Gdje je *c* kompleksan broj. Sada se originalne realne vrijednosti signala mogu izvući iz komponenata od *c* na idući način:

$$a = R(c) \qquad\qquad b = I(c)$$

Ova optimizacija ima značajan utjecaj, jer većina podataka potrebna za renderiranje dolazi u obliku para Hermitskih spektara.

Implementacija brze Fourijerove transformacije:

```
#version 430

#define PI                3.1415926535897932
#define TWO_PI  6.2831853071795864
#define DISP_MAP_SIZE 512
#define LOG2_DISP_MAP_SIZE 9

layout (rg32f, binding = 0) uniform readonly image2D readbuff;
layout (rg32f, binding = 1) uniform writeonly image2D writebuff;

vec2 ComplexMul(vec2 z, vec2 w) {
        return vec2(z.x * w.x - z.y * w.y, z.y * w.x + z.x * w.y);
}

shared vec2 pingpong[2][DISP_MAP_SIZE];

layout (local_size_x = DISP_MAP_SIZE) in;
void main()
{
        const float N = float(DISP_MAP_SIZE);

        int z = int(gl_WorkGroupID.x);
        int x = int(gl_LocalInvocationID.x);

        // STEP 1: load row/column and reorder
        int nj = (bitfieldReverse(x) >> (32 - LOG2_DISP_MAP_SIZE)) & (DISP_MAP_SIZE -
1);
        pingpong[0][nj] = imageLoad(readbuff, ivec2(z, x)).rg;

        barrier();

        // STEP 2: perform butterfly passes
        int src = 0;

        for (int s = 1; s <= LOG2_DISP_MAP_SIZE; ++s) {
                int m = 1 << s;                                 // butterfly group
height
                int mh = m >> 1;                        // butterfly group half
height

                int k = (x * (DISP_MAP_SIZE / m)) & (DISP_MAP_SIZE - 1);
                int i = (x & ~(m - 1));                 // butterfly group starting
offset
                int j = (x & (mh - 1));                 // butterfly index in group
```

```
41.
42.                 // twiddle factor W_N^k
43.                 float theta = (TWO_PI * float(k)) / N;
44.                 vec2 W_N_k = vec2(cos(theta), sin(theta));
45.
46.                 vec2 input1 = pingpong[src][i + j + mh];
47.                 vec2 input2 = pingpong[src][i + j];
48.
49.                 src = 1 - src;
50.                 pingpong[src][x] = input2 + ComplexMul(W_N_k, input1);
51.
52.                 barrier();
53.         }
54.
55.         // STEP 3: write output
56.         vec2 result = pingpong[src][x];
57.         imageStore(writebuff, ivec2(x, z), vec4(result, 0.0, 1.0));
58.
60. }
61.
```

3.4 Nagibi i pomaci

U ovom trenutku moguće je izračunati visinu površine primjenom inverzne diskretne Fourierove transformacije na generiranom spektru. Za potrebe sjenčanja također treba pronaći vektore nagiba površine kako bi se mogle izračunati normale površine. Najjednostavniji način za izračun nagiba je korištenje konačnih razlika u prostornoj domeni površine. Takav pristup je efikasan u pogledu memorije i računanja, ali može nedostajati kvalitete jer može biti loša aproksimacija nagiba valova s malim valnim duljinama. Preciznije vektore nagiba može se dobiti primjenom spektralne derivacije. Spektralna derivacija omogućava pronalaženje derivacija funkcije pomoću Fourierove transformacije.

Ako opći slučaj diskretne inverzne Fourierove transformacije izgleda ovako:

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

Tada n-ta derivacija glasi:

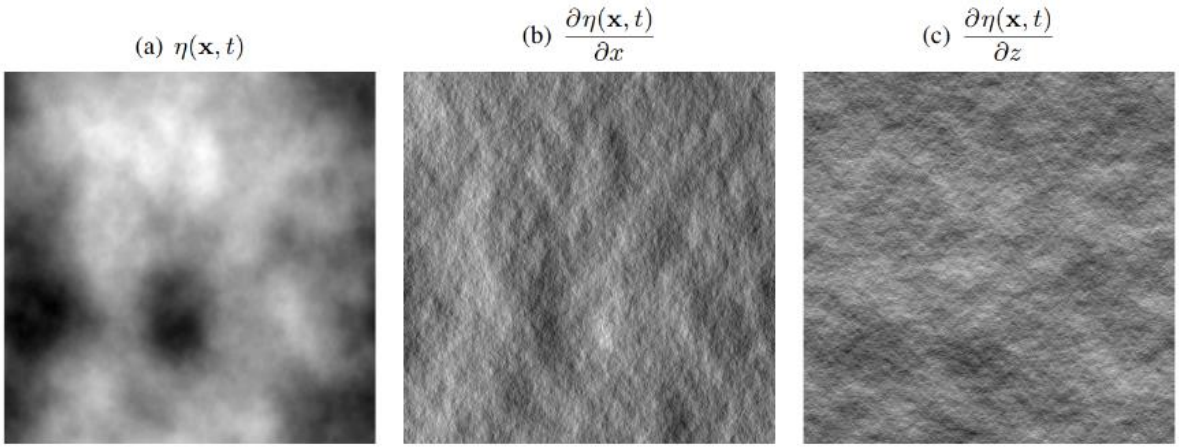
$$\frac{d^n f(\mathbf{x})}{dx^n} = \sum_{\mathbf{k}} (i\mathbf{k})^n g(k) * e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x})} \qquad (3.7)$$

Sada se nagibi mogu izračunati na sljedeći način:

Budući da je potrebno naći nagibe diskretnih točaka površine, potrebno je izračunati parcijalne derivacije prvog reda funkcije elevacije. Dvodimenzionalni vektor nagiba može se dobiti ovako:

$$\mathbf{s}(\mathbf{x}, t) = [\frac{\partial f(\mathbf{x})}{\partial x}, \frac{\partial f(\mathbf{x})}{\partial z}] \qquad (3.8)$$

Dva spektra koja konstituiraju nagib površine u domeni valnih vektora su Hermitska. Stoga je moguće primijeniti optimizaciju 3.6 koja omogućava računanje obje komponente nagiba sa samo jednom inverznom Fourierovom transformacijom.



Slika 3.3 Vizualizacija Fourierovih transformacija nagiba

Budući da se u ovom radu koristi desni kartezijanski sustav, gdje Y-os pokazuje u smjeru suprotnom od gravitacijske sile, vektor nagiba leži u XZ ravnini. Tada se vektor normale može dobiti idućim računom:

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

Gdje su s_x i s_z komponente vektora nagiba dobivenog sa 3.5.

Primjer računskog sjenčara za izračunavanje gradijenata.

```
#version 430

#define DISP_MAP_SIZE 512
#define INV_TILE_SIZE 25.6
#define TILE_SIZE_X2 0.15625

layout (rgba32f, binding = 0) uniform readonly image2D displacement;
layout (rgba16f, binding = 1) uniform writeonly image2D gradients;

layout (local_size_x = 16, local_size_y = 16) in;
void main()
{
```

```
13.       ivec2 loc = ivec2(gl_GlobalInvocationID.xy);
14.
15.       // gradient
16.       ivec2 left                                = (loc - ivec2(1, 0)) & (DISP_MAP_SIZE
- 1);
17.       ivec2 right                               = (loc + ivec2(1, 0)) & (DISP_MAP_SIZE
- 1);
18.       ivec2 bottom                  = (loc - ivec2(0, 1)) & (DISP_MAP_SIZE - 1);
19.       ivec2 top                     = (loc + ivec2(0, 1)) & (DISP_MAP_SIZE - 1);
20.
21.       vec3 disp_left                = imageLoad(displacement, left).xyz;
22.       vec3 disp_right               = imageLoad(displacement, right).xyz;
23.       vec3 disp_bottom    = imageLoad(displacement, bottom).xyz;
24.       vec3 disp_top                 = imageLoad(displacement, top).xyz;
25.
26.       vec2 gradient                 = vec2(disp_left.y - disp_right.y, disp_bottom.y
- disp_top.y);
27.
28.       // Jacobian
29.       vec2 dDx = (disp_right.xz - disp_left.xz) * INV_TILE_SIZE;
30.       vec2 dDy = (disp_top.xz - disp_bottom.xz) * INV_TILE_SIZE;
31.
32.       float J = (1.0 + dDx.x) * (1.0 + dDy.y) - dDx.y * dDy.x;
33.
34.       // NOTE: normals are in tangent space for now
35.       imageStore(gradients, loc, vec4(gradient, TILE_SIZE_X2, J));
36. }
37.
```

3.4.1 Pomaci

Valovi generirani dosad prezentiranim metodama imaju zaobljene brijegove i dolove, što je uobičajeno za dobrog vremena. Također bi poželjno bilo moći simulirati valove za vrijeme nepovoljnih vremenskih uvjeta kao što su jaki vjetrovi ili oluje. Za takvog vremena brjegovi su oštri, a dolovi poravnati. Tessendord[1] opisuje metodu zasnovanu na Fourierovim transformacijama kojom se postiže takav učinak, koja se zove „algoritam izlomljenih valova“ (engl. choppy wave algorithm). Na slici 3.5 primjećuje sa razlika površine sa i bez primjene algoritma izlomljenih valova. Na svaku točku plohe primjenjuje se pomak u XZ ravnini. Taj pomak ponovno se računa pomoću inverznih brzih Fourierovih transformacija spektra *h*:

$$\mathbf{D}(\mathbf{x}, t) = [D_x(x, t), D_z(x, t)] \tag{3.9}$$

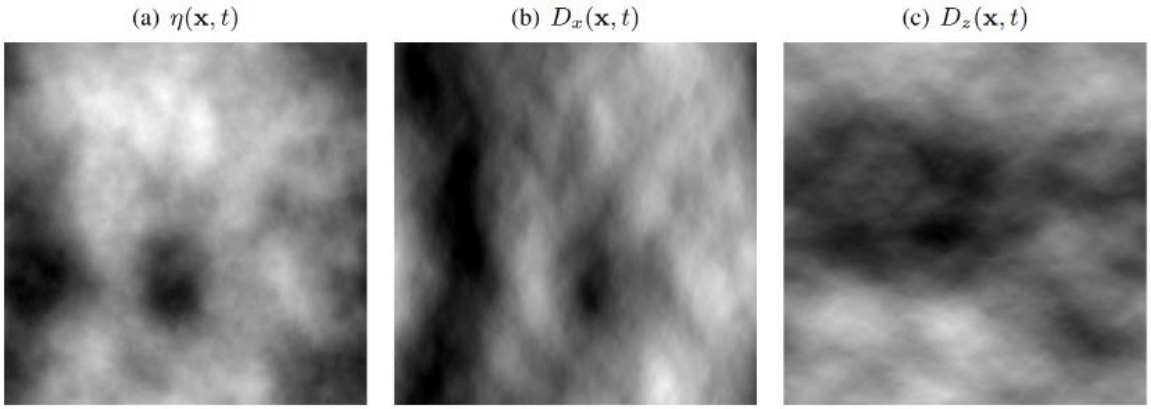
$$D_x(\mathbf{x}, \mathrm{t}) = \sum_{\mathbf{k}} -i \frac{k_x}{k} h(\mathbf{k}, t) * e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x})} \tag{3.10}$$

$$D_z(\mathbf{x}, \mathrm{t}) = \sum_{\mathbf{k}} -i \frac{k_z}{k} h(\mathbf{k}, t) * e^{i(\mathbf{k}^{\mathbf{T}}\mathbf{x})} \tag{3.11}$$

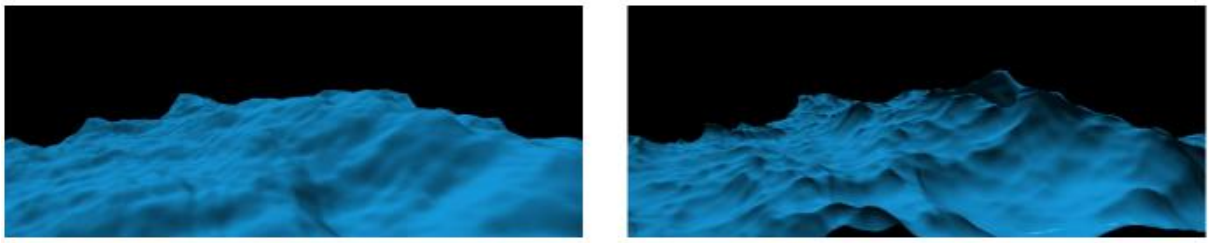
Sada za dane pomake, vrh u trodimenzionalnom prostoru može se zapisati ovako:

$$\mathbf{V} = [\mathrm{x} + u\mathrm{Dx}, \eta(\mathbf{x}, t),\ \mathrm{z} + u\mathrm{Dz}\ ] \qquad (3.12)$$

Gdje je $u \in \mathrm{R}$ hiperparametar kojim se određuje skala pomaka. Sve dok je $u < 0$ postiže se traženi učinak, a kada taj uvjet ne vrijedi postiže se kontra-efekt, gdje se valovi doimaju još zaobljeniji.



Slika 3.4 Vizualizacija pomaka po svim dimenzijama



Slika 3.5 Primjer efekta izlomljeniih valova

Sjenčar za računanje pomaka.

```
#version 430

#define DISP_MAP_SIZE 512
#define INV_TILE_SIZE 25.6
#define TILE_SIZE_X2 0.15625

layout (rgba32f, binding = 0) uniform readonly image2D displacement;
layout (rgba16f, binding = 1) uniform writeonly image2D gradients;

layout (local_size_x = 16, local_size_y = 16) in;
void main()
{
	ivec2 loc = ivec2(gl_GlobalInvocationID.xy);

	// gradient
	ivec2 left                              = (loc - ivec2(1, 0)) & (DISP_MAP_SIZE - 1);
	ivec2 right                             = (loc + ivec2(1, 0)) & (DISP_MAP_SIZE - 1);
	ivec2 bottom                    = (loc - ivec2(0, 1)) & (DISP_MAP_SIZE - 1);
	ivec2 top                       = (loc + ivec2(0, 1)) & (DISP_MAP_SIZE - 1);

	vec3 disp_left                  = imageLoad(displacement, left).xyz;
	vec3 disp_right                 = imageLoad(displacement, right).xyz;
	vec3 disp_bottom    = imageLoad(displacement, bottom).xyz;
	vec3 disp_top                   = imageLoad(displacement, top).xyz;

	vec2 gradient                   = vec2(disp_left.y - disp_right.y, disp_bottom.y - disp_top.y);

	// Jacobian
	vec2 dDx = (disp_right.xz - disp_left.xz) * INV_TILE_SIZE;
	vec2 dDy = (disp_top.xz - disp_bottom.xz) * INV_TILE_SIZE;

	float J = (1.0 + dDx.x) * (1.0 + dDy.y) - dDx.y * dDy.x;

	// NOTE: normals are in tangent space for now
	imageStore(gradients, loc, vec4(gradient, TILE_SIZE_X2, J));
}

```

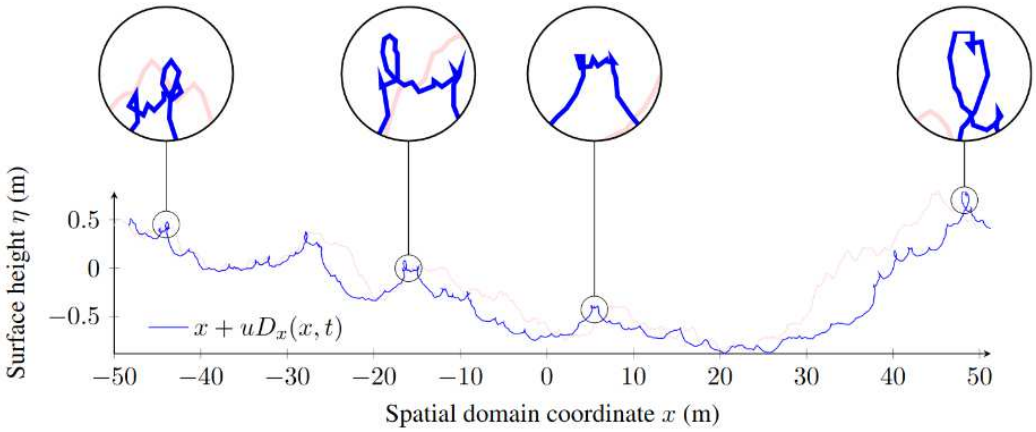
3.4.2 Nagibi nakon pomaka

Primjena pomaka na točke rešetke također utječe na nagibe površine, koje je dakle potrebno preračunati u novim posmaknutim točkama.

Prema Dupuy-u i Brunetonu [16] nagib površine u točki dobiva se na sljedeći način:

$$\mathbf{s}(\mathbf{x},t) = \left[\frac{\frac{d\eta(\mathbf{x},t)}{dt}}{1+u\frac{dD_x(\mathbf{x},t)}{dx}}, \quad \frac{\frac{d\eta(\mathbf{x},t)}{dt}}{1+u\frac{dD_z(\mathbf{x},t)}{dz}}\right]$$

Algoritam izlomljenih valova ne dolazi bez nedostataka. Pomaci u određenim slučajevima mogu biti dovoljno veliki da dolazi do preklapanja točaka, čije je posljedica da normala površine između dvije točke koje se presijecaju ima usmjerenje prema dolje. Slika 3.6 vizualno predočuje taj efekt.



Slika 3.6

Smanjenjem parametra „*u*“ može se spriječiti takva situacija, no ta se pojava može iskoristiti u pozitivne svrhe. Mjesta gdje se dvije točke presijecaju mogu se gledati kao mjesta gdje se lome valovi, te time postaju potencijalne lokacije za generiranje morske pjene. Tako bi trebalo pronaći vremenski učinkovit način za testiranje takvih pojava. Tessendorf preporučuje računanje determinante Jacobijeve matrice transformacije pomakom.

Ako funkcija pomaka glasi:

$$g(X,t) = x + uD(x,t)$$

tada bi jakobijan J glasio :

$$\mathbf{J}(x,t) = \begin{bmatrix} Jxx & Jxz \\ Jzx & Jzz \end{bmatrix}$$

Sada determinanta ove matrice glasi:

$$det(J(x,t)) = JxxJzz - JzxJxz$$

Kada je ona negativna, točke površine se preklapaju. U idućem poglavlju o sjenčanju objašnjeno je kako točno pomoću determinante jakobijana generirati morsku pjenu.

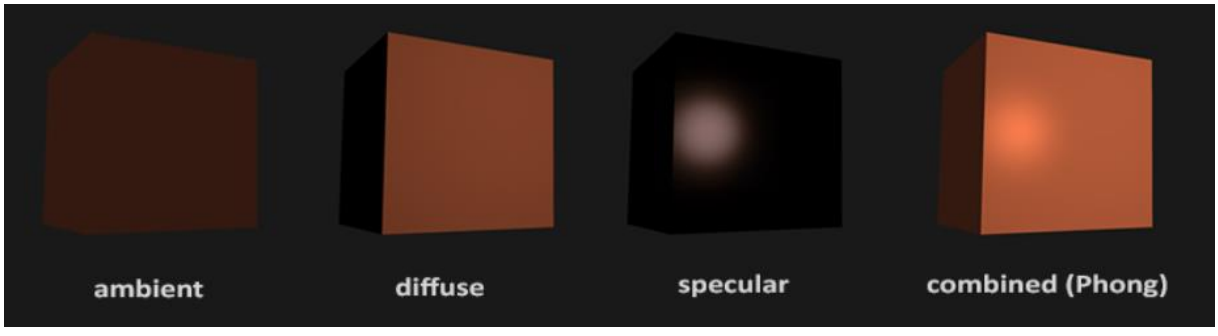
4. Sjenčanje

U stvarnom svijetu boje mogu imati bilo koju poznatu vrijednost, pri čemu svaki objekt ima vlastitu boju ili boje. U digitalnom svijetu potrebno je mapirati (beskonačne) stvarne boje na (ograničene) digitalne vrijednosti, pa stoga nije sve boje iz stvarnog svijeta moguće prikazati digitalno. Boje su digitalno predstavljene pomoću crvene, zelene i plave komponente, obično skraćeno kao RGB. Kombiniranjem samo te 3 vrijednosti unutar raspona [0,1], može se prikazati gotovo svaka boja. Na primjer, za dobivanje boje koralja definira se vektor boje kao

```
vec3 coral(1.0f, 0.5f, 0.31f);
```

Boja objekta koja se vidi u stvarnom životu nije stvarna boja koju on ima, već boja koja se reflektira s objekta. Na primjer, sunčeva svjetlost percipira se kao bijela svjetlost koja je kombinacija mnogih različitih boja. Kada bi se ta bijela svjetlost usmjerila na plavu igračku, ona bi apsorbirala sve pod-boje bijele svjetlosti osim plave. Budući da igračka ne apsorbira plavi dio svjetlosti, on se reflektira. Ta reflektirana svjetlost ulazi u oko, zbog čega se stvara dojam da igračka ima plavu boju. Ova pravila refleksije boja izravno se primjenjuju i u području računalne grafike

Osvjetljenje u stvarnom svijetu izuzetno je složeno i ovisi o previše faktora, nešto što u računalnom smislu nije moguće priuštiti računati s ograničenom računalnom snagom kojom se raspolaže. Stoga se osvjetljenje grafici temelji na aproksimacijama stvarnosti korištenjem pojednostavljenih modela koji su mnogo lakši za procesiranje i izgledaju relativno slično. Ti modeli osvjetljenja temelje se na fizici svjetlosti . Jedan od tih modela naziva se Phongov model osvjetljenja. Glavni građevni blokovi Phongovog modela osvjetljenja sastoje se od 3 komponente: ambijentalnog, difuznog i reflektivnog (spekularnog) osvjetljenja. U nastavku je prikazano kako te komponente osvjetljenja izgledaju zasebno i u kombinaciji.



Slika 4.1 Doprinos komponenata u Phongovom modelu

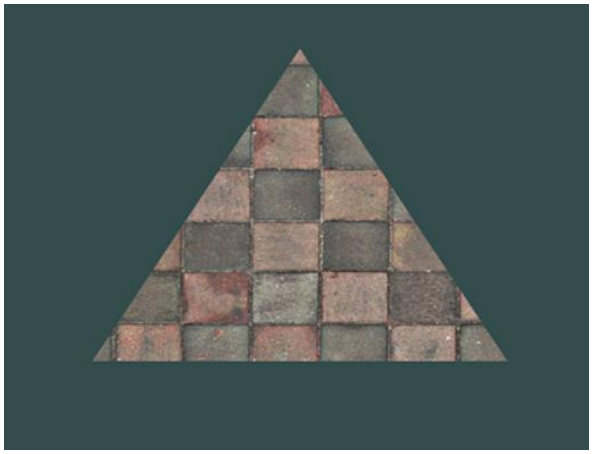
- Ambijentalno svjetlo – doprinos okoline tj. ambijenta
- Difuzno svjetlo – simulira utjecaj svjetla na objekt i vizualno najviše doprinosi, što je normala površine usmjerenija prema izvoru svjetlosti, to je površina svjetlija
- Spekularno svjetlo – svjetla točka koja imitira odsjaj. Bliže je boji izvora svjetlosti, a ovisno o materijalu objekta može se podešavati hiperparametar koji proizvodi manji ili veći odsjaj

Međutim, ovakav jednostavan pristup za sjenčanje vode nije dovoljno dobar. Pruža „plastičan“ izgled, a voda je donekle prozirno tijelo i kao takvo za postizanje realizma treba se povećati razina kompleksnosti.

U nastavku su obrađene tehnike koje se koriste za postizanje realističnosti.

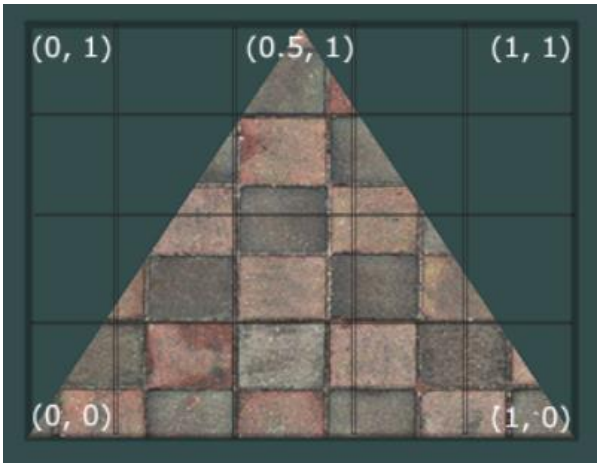
4.2 Mapa okoliša

Tekstura, u kontekstu računalne grafike, najčešće je 2D slika. Koriste se za dodavanje detalja objektima. Primjerice, ako je cilj prikazati trokut tako da izgleda kao da je od cigli, moguće je sa teksture uzorkovati piksele te ih „nalijepiti“ na trokut.



Slika 4.2 Primjer teksture

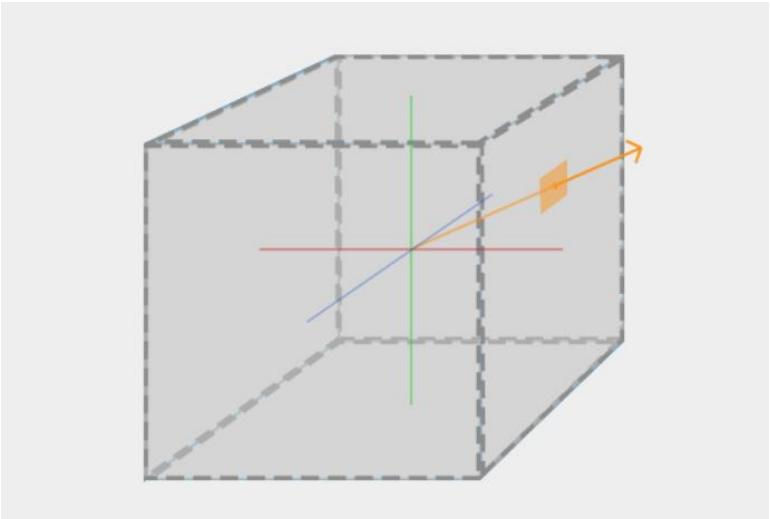
Pikseli teksture uzorkuju se od 0 do 1 po svakoj dimenziji, pa bi se svakom vrhu trokuta dodijelile koordinate za uzorkovanje kao na slici 4.3



Slika 4.3 Uzorkovanje teksture

Mapu okoliša može se konstruirati kao 6 individualnih 2D tekstura, tako da svaka od njih formira jednu stranicu kocke.

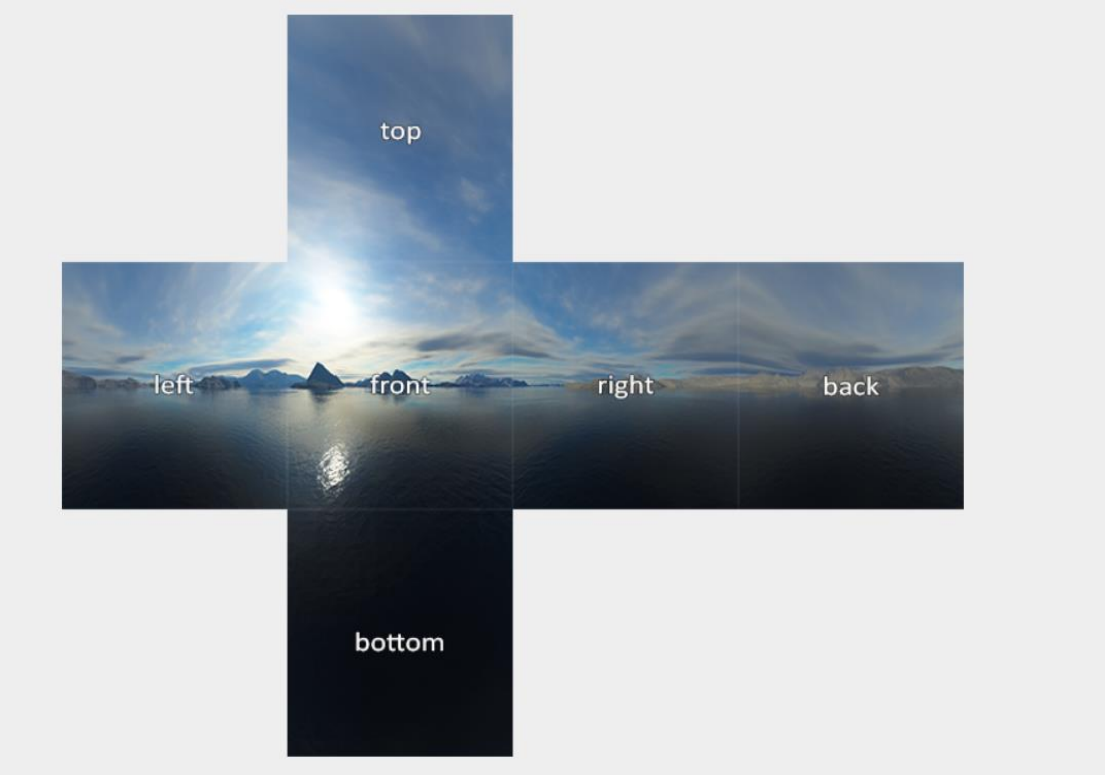
Tada se očište promatra kao središte kocke, a smjer od očišta do gledišta se koristi kao vektor za uzorkovanje iz teksture. Slika 5.4 opisuje ovaj postupak.



Slika 4.4 Uzorkovanje iz kubne mape

One su povoljne za ilustraciju nebeskih svodova. Također, pošto ono što se uzorkuje ovisi samo o smjeru gledanja, ali ne i poziciji očišta, stvara se dojam kao da je kubna mapa beskonačno velika, te pomicanje naprijed-nazad-lijevo-desno-gore-dolje nema ulogu u onome što se uzorkuje, što je upravo ono što je i cilj kod prikaza nebeskog svoda.

Slika 4.5 dobar je primjer tekstura koje čine kubnu mapu.



Slika 4.5 Teksture koje čine kubnu mapu

Slika 4.6 primjer je scene dobivene korištenjem kubne mape.

Slika 4.6

Kako je ocean reflektivno tijelo i često je vizualno uparen sa nebeskim svodom, ova tehnika iznimno je korisna za prikaz odraza neba od vode.

Kao komponentu sjenčanja vode, dodaje se uzorak iz kubne mape tj. mape okoliša. Uzorak se uzima tako da, umjesto smjera od očišta do gledišta, se koristi smjer refleksije tog vektora od točke na površini.

Slika 4.7 Korištenje mape okoliša kao komponente za sjenčanje površine

GLSL sjenčar fragmenata – primjer uzorkovanja iz mape okoliša + choppy wave.

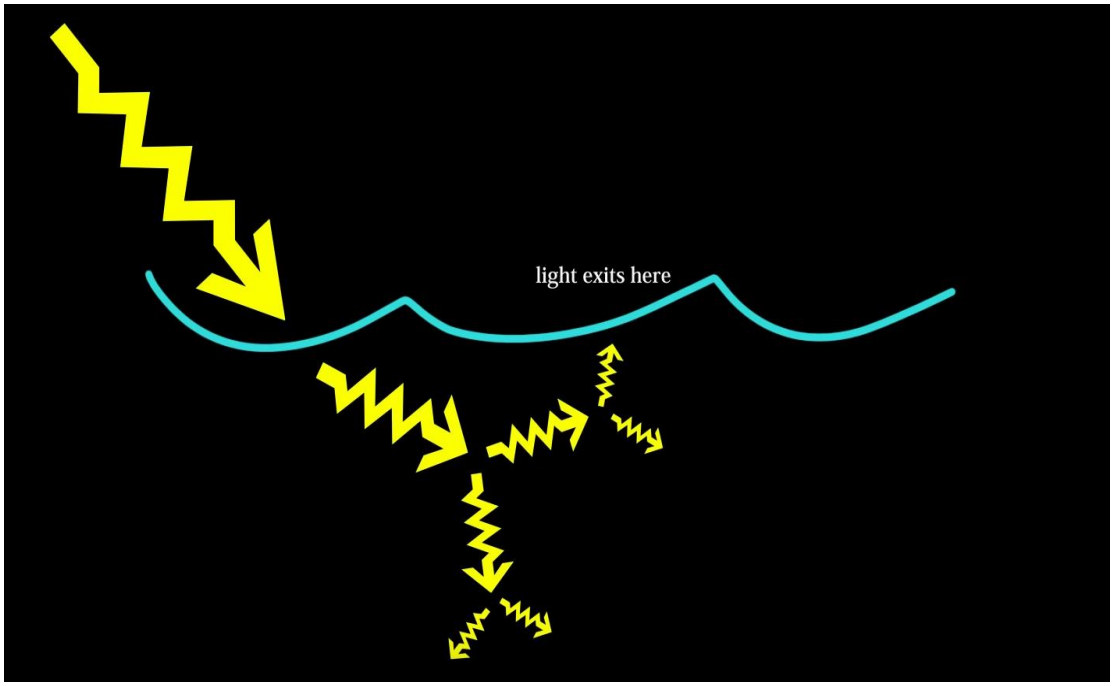
```
        vec4 slope = texture(gradients, texCoord);
        vec4 grad = slope;
        grad.xzy = normalize(vec3(
      - ( slope.x / (1.0f + choppy * slope.z) ),
      1.0f,
      - ( slope.y / (1.0f + choppy * slope.w) )
  ));
        grad.w = slope.w;

                  vec3 viewDirr = cameraPos - fragPos.xyz;
        float dist = length(viewDirr.xz);

        vec3 n = normalize(grad.xzy);
        vec3 v = normalize(viewDirr);
        vec3 l = reflect(-v, n);
        vec3 refl = texture(envmap, l).rgb;
```

4.2 Raspršenje

Kako je difuzno svjetlo namijenjeno za opis neprozirnih objekata, nije primjereno za korištenje pri sjenčanju vode. Tako da se ta komponenta zamjenjuje sa komponentom *raspršenja* koja opisuje doprinos zraka svjetlosti koje se lome prilikom prolaska kroz vodu i izlaze nazad kroz površinu.



Slika 4.8 Podvodna refrakcija svjetlosti

Pratiti projekciju putanje svih zraka nakon svakog raspršivanja je ekstremno kompleksno, gotovo nemoguće te računalno zahtjevno. Nasreću, može se postići dovoljno dobra imitacija koja postiže takav učinak, a lako je izvediva. U tome pomaže opažanje da zrake koje izlaze nazad iz vode uvelike dolaze kada je kut upada svjetlosti jako malen i kada su vrhovi valova jako visoki, jer je tada veća vjerojatnost da će se dio zraka raspršiti prema vani.

$$L_{scatter} = (k_1 H\langle \omega_i \cdot -\omega_o \rangle^4 (0.5 - 0.5(\omega_i \cdot \omega_n))^3 + k_2 \langle \omega_o \cdot \omega_n \rangle^2) C_{ss} L_{sun} \cdot \frac{1}{(1 + \Lambda(\omega_i))}$$

$$L_{scatter} \mathrel{+}= k_3 \langle \omega_i \cdot w_n \rangle C_{ss} L_{sun} + k_4 P_f C_f L_{sun}$$

Slika 4.9 Primjer komponente raspršenja

Tako je u raspršeno svjetlo ubrojano više komponenti:

- Raspršenost pri visokom valu
- Kut između normale i kamere
- Kut između normale i svjetla
- Ambijentalno svjetlo

Svaka od ovih komponenata pomnožena je sa bojom sunca, bojom raspršivanja(obično plava), te ambijentalnom bojom.

4.3 Morska pjena

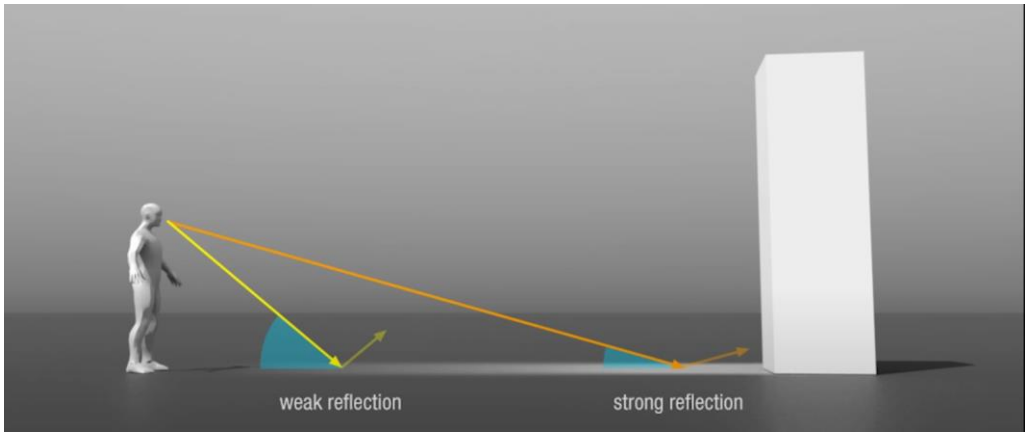
U poglavlju 3 obrađen je potencijalni način na koji se može detektirani mjesto na kojem može doći do nastajanja pjene. Mjesta na kojima je determinanta jakobijana funkcije posmaka manja od zadane vrijednosti može se smatrati lokacijama za generiranje morske pjene. Ta zadana vrijednost najčešće je 0, međutim, ukoliko je cilj manipulirati količinu morske pjene koja se pojavljuje, granica se može pomicati gore ili dolje.

Kako bi ovaj učinak došao do izražaja potrebno je pjenu , umjesto samo prikazati u datom trenutku, akumulirati. Da bi se to izvelo, potrebno je pratiti simulaciju pjene kroz vrijeme spremajući podatke u teksturu dimenzija jednakih dimenzijama polja. Kada jakobijan za određeni vrh bude negativan, dodaje se malo pjene na odgovarajućem mjestu u strukturi podataka. U implementacijskom smislu to bi značilo da će se boje tih vrhova, koji su originalno primjerice plave boje, #0000FF , postaviti na bijelu, #FFFFFF , ukoliko matrica nalaže da se na tom mjestu nalazi pjena.

Pri učitavanju svakog okvira se, uz dodavanje pjene na određenim mjestima, na teksturu primjenjuje funkcija eksponencijalnog pada (interpolira se od #FFFFFF do #0000FF), čime se postiže učinak da pjena iščezava kroz vrijeme, umjesto momentalno idući okvir.

4.2 Fresnelov efekt

Ovaj efekt opisuje vezu između kuta upada svjetlosti i reflektivnosti odraza. Ukratko – površine su puno refleksivnije kad su promatrane pod manjim kutevima. To je svakodnevna pojava u prirodi i vrijedi za sve površine.



Slika 4.10 Fresnelov efekt

Izračun fresnel komponente u sjenčaru:

```
1.                          // Schlick Fresnel
2.                          float3 fresnelNormal = normal;
3.                          fresnelNormal.xz *= _FresnelNormalStrength;
4.                          fresnelNormal = normalize(fresnelNormal);
5.                          float base = 1 - dot(viewDir, fresnelNormal);
6.                          float exponential = pow(base, _FresnelShininess);
7.                          float R = exponential + _FresnelBias * (1.0f
exponential);
8.                          R *= _FresnelStrength;
9.                          float3 fresnel = _FresnelColor * R;
```

4.4 Popločavanje

Iako je simuliranje vode kao linearne kombinacije sinusnih valova uz pomoć brze Fourierove transformacije brzo i efikasno, postoji problem.

Slika 4.11 Primjer popločavanja

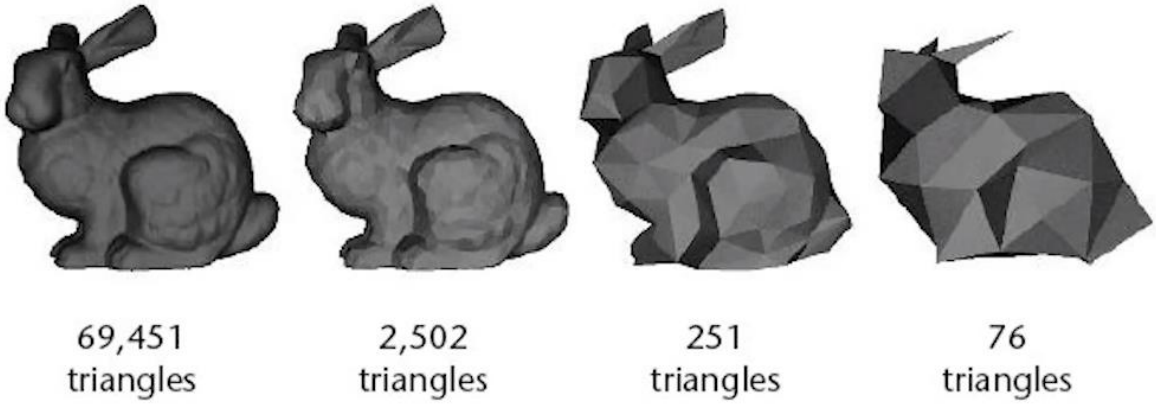
Gledajući sliku 4.11 vidi se da, iz ptičje perspektive, zbog periodične prirode sinusnih valova, se stvara efekt popločavanja, koji ostavlja jako neprirodan dojam. Ovaj problem za sada nema rješenje u računalnoj grafici, međutim postoje metode kojima se može mitigirati.

Slika 4.12 prikazuje primjer LOD sustava (Level-Of-Detail). Cilj LOD sustava je, imitirajući ljudsko oko, smanjivati rezoluciju objekata na temelju udaljenosti. Ovo ima višestruke koristi.

Uz ublažavanje efekta popločavanja, potencijalno se rasterećuje grafička kartica, koja manje vremena troši na prikaz daljih objekata.

Dodatan način mitigiranja efekta popločavanje je „naslagivanje“ više simulacije jedne na drugu. Generiranje više spektara različitih rezolucija nije računalno skupo, a odlaže

efekt popločavanja za područja veličine najmanjeg zajedničkog višekratnika danih rezolucija.



Slika 4.12 Primjer LOD sustava

Zaključak

U ovom radu istražene su metode simulacije oceanskih valova u računalnoj grafici, s posebnim naglaskom na fizikalno utemeljene modele i optimizacije za prikaz u stvarnom vremenu. Raspravljeni su teorijski temelji generiranja vjerodostojnog modela, temeljenog na linearnog valnoj teoriji i relaciji disperzije.

Kroz analizu spektralnih modela, poput Pierson-Moskowitz i JONSWAP spektra, se na temelju empirijski izmjerenih podataka generira domena valnih vektora koji sudjeluju u oblikovanju morske površine. Primjenom optimizacija na implementaciju Fourierove transformacije omogućeno je prikazivanje simulacije površine u stvarnom vremenu. Korištenjem nasumičnog uzorkovanja spektra postignut je realističan efekt valova, dok je dodatkom pomaka i nagiba poboljšana razina detalja.

Također, istraženi su osnovni svjetlosni modeli, kao Phongov model, te je objašnjeno zašto nije primjenjiv u ovoj specifičnoj situaciji. Dane su tehnike za modifikaciju svjetlosnog modela za postizanje realizma, kao što su fresnelov efekt, korištenje mape okoliša, komponente raspršenja te generiranje morske pjene. Napravljen je osvrt izazov u vizualizaciji koji se pojavljuje u vidu efekta popločavanja. Generiranje morske pjene izvedeno je analizom determinante Jacobijeve matrice, što omogućuje pojavu pjene na područjima s visokim turbulentnim valovima. Problem popločavanja riješen je primjenom više spektralnih slojeva različitih rezolucija, čime se povećava vizualni realizam te razvojem LOD sustava.

Navedeni elementi osiguravaju visokokvalitetnu grafiku u stvarnom vremenu, pružajući temelje za primjenu u videoigrama i simulacijama. Rad zaključuje kako su fizikalno utemeljeni modeli, kombinirani s optimizacijama za GPU, ključni za postizanje uvjerljivih simulacija valova, dok dodatni efekti doprinose realizmu i atraktivnosti završnog prikaza.

Literatura

- [1] G.B. Airy. Tides and Waves: Extracted from the Encyclopaedia Metropolitana, Tom. V Pag. 241 - 396. William Clowes and Sons, 1845
- [2] B. Kinsman. Wind Waves: Their Generation and Propagation on the Ocean Surface. Dover Phoenix Editions. Dover Publications, 2002.
- [3] G.K. Batchelor. An Introduction to Fluid Dynamics. Cambridge Mathematical Library. Cambridge University Press, 2000.
- [4] Gerhard Neumann and Willard J Pierson Jr. Principles of physical oceanography. Englewood Cliffs, NJ (USA) Prentice Hall, 1966
- [5] G. Grimmett and D. Stirzaker. Probability and Random Processes. Probability and Random Processes. OUP Oxford, 2001.
- [6] R.N. Bracewell. The Fourier Transform and Its Applications. Electrical engineering series. McGraw-Hill Higher Education, 2000.
- [7] G.J. Komen, L. Cavaleri, M. Donelan, K. Hasselmann, S. Hasselmann, and P.A.E.M. Janssen. Dynamics and Modelling of Ocean Waves. Cambridge University Press, 1996.
- [8] https://www.cg.tuwien.ac.at/research/publications/2018/GAMPER-2018-OSG/GAMPER-2018-OSG-thesis.pdf#figure.3.6
- [9] Tessendorf, Jerry. (2001). Simulating Ocean Water. SIG-GRAPH'99 Course Note.
- [10] Willard J. Pierson and Lionel Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. J. Geophys. Res., 69(24), December 1964
- [11] K. Hasselman, T. P. Barnett, E. Bouws, D. E. Carlson, and P. Hasselmann. Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap). Deutsche Hydrographische Zeitschrift, 8(12), 1973
- [12] O. M. Phillips. The equilibrium range in the spectrum of wind-generated waves. Journal of Fluid Mechanics, 4:426–434, 8 1958.
- [13] O. M. Phillips. Spectral and statistical properties of the equilibrium range in wind-generated gravity waves. J. Fluid Mech., 156:505–531, 1985

- [14] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. Mathematics of Computation, 19(90):297–301, 1965

- [15] Matteo Frigo and Steven G. Johnson. FFTW. Massachusetts Institute of Technology, 2017. Last Accessed: 2017-08-21

- [16] Jonathan Dupuy and Eric Bruneton. Real-time animation and rendering of ocean whitecaps. In SIGGRAPH Asia 2012 Technical Briefs, SA ’12, pages 15:1–15:3. ACM, 2012

Sažetak

Ovaj rad istražuje simulaciju oceanskih valova u računalnoj grafici s fokusom na fizikalno utemeljene modele i optimizacije za prikaz u stvarnom vremenu. Korištenjem Fourierove transformacije i spektralnih modela, poput Pierson-Moskowitz i JONSWAP spektra, postignuti su realistični efekti valova. Problem popločavanja riješen je slojevitim spektralnim uzorkovanjem, dok je generiranje morske pjene omogućeno analizom Jacobijeve matrice. Uvođenje Fresnelovog efekta i raspršenja svjetlosti dodatno je poboljšalo vizualni realizam. Zaključuje se da kombinacija fizikalnih modela i GPU optimizacija omogućuje stvaranje uvjerljivih simulacija valova za primjenu u videoigrama i simulacijama.

Summary

This thesis explores the simulation of ocean waves in computer graphics, focusing on physically-based models and real-time rendering optimizations. Using the Fourier Transform and spectral models such as the Pierson-Moskowitz and JONSWAP spectra, realistic wave effects were achieved. Tiling issues were addressed with layered spectral sampling, while foam generation was enabled through Jacobian matrix analysis. The introduction of the Fresnel effect and light scattering further enhanced visual realism. The study concludes that the combination of physically-based models and GPU optimizations enables the creation of convincing wave simulations for applications in video games and simulations.