Universidade Federal Fluminense – Instituto de Computação

Alex Vanderlei Salgado

Simulação visual em tempo real de ondas oceânicas utilizando a GPU

Dissertação de Mestrado

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Mestre. Área de concentração: Computação Visual e Interfaces.

Orientadora: Prof[a]. Aura Conci

Niterói, dezembro de 2006

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

S164 Salgado, Alex Vanderlei.

Simulação visual em tempo real de ondas oceânicas utilizando a GPU / Alex Vanderlei Salgado. – Niterói, RJ : [s.n.], 2006.
73 f.

Orientador: Aura Conci.

Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, 2006.

1. Computação visual. 2. Ondas oceânicas. 3. Fenômeno natural. I. Título.

CDD 006.6

Universidade Federal Fluminense – Instituto de Computação

Alex Vanderlei Salgado

Simulação visual em tempo real de ondas oceânicas utilizando a GPU

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para obtenção do título de Mestre. Área de concentração: Computação Visual e Interfaces.

Banca Examinadora

Prof[a]. Aura Conci
Universidade Federal Fluminense

Prof. Esteban Walter Gonzalez Clua
Universidade Federal Fluminense

Prof. Waldemar Celes Filho
PUC-Rio

Niterói, 22 de dezembro de 2006

À memória de **Raimunda**, minha avó

Aos meus pais

Agradecimentos

À Deus, onde sempre depositei minhas esperanças e de onde sempre tirei forças para não desistir deste sonho.

À Dayse, pelo amor, companheirismo, paciência e incentivo, acompanhando ao meu lado toda a minha trajetória e as madrugadas em frente ao computador, até aprendendo como se formam as ondas do mar.

À minha família, pais e irmãos, que mesmo distante é sempre um porto seguro e com simples palavras e gestos me confortaram trazendo uma inspiração a mais para continuar em frente.

À Aura, minha orientadora e amiga sempre incentivando, orientando e tornando viável o sonho da realização do mestrado.

Aos meus amigos de trabalho, da faculdade, do convívio do dia-a-dia, aos antigos e aos novos, aos que estão próximos e aos que estão distantes, que compreenderam que é necessário um certo isolamento social para que este trabalho acontecesse, e que com incentivo e idéias também contribuíram para esta dissertação.

Ao mar e as suas ondas, minha eterna fonte de inspiração.

É melhor atirar-se em luta, em busca de dias melhores, do que permanecer estático como os pobres de espírito, que não lutaram, mas também não venceram. Que não conheceram a glória de ressurgir dos escombros. Esses pobres de espírito, ao final de sua jornada na Terra, não agradecem à Deus por terem vivido, mas desculpam-se diante dele, por simplesmente, haverem passado pela vida.

(Bob Marley)

Resumo

A modelagem de fenômenos naturais é umas das tarefas mais desafiadoras da computação gráfica. Dentro deste desafio está a simulação das ondas oceânicas. Com o processamento gráfico da GPU, é possível utilizar técnicas avançadas de rendering em aplicações em tempo real com grande realismo. Este trabalho inicia uma linha de pesquisa de modelagem e renderização de fenômenos naturais em tempo real na área de Computação Visual e Interfaces da pós-graduação em computação do Instituto de Computação da UFF. Esta dissertação busca simular o comportamento das ondas oceânicas processando todo cálculo geométrico e renderização na GPU. A forma da onda é definida pela equação de Gerstner onde são simulados os movimentos das partículas de água. É possível considerar a representação tanto em águas profundas quanto em águas rasas. A modelagem também simula como as ondas se quebram ao aproximar-se da costa e como sofrem influência da topologia do fundo do mar (refração de ondas), alterando adequadamente suas propriedades geométricas. A renderização implementada usa uma combinação de técnicas avançadas de renderização em tempo real utilizando a GPU que vão da reflexão por bump mapping no espaço da tangente e environment map a reflexão de Fresnel e HDR. Como mostrado nos exemplos deste trabalho estas técnicas criam uma visualização da cena bem realista e bastante adequada.

Palavras-chave

GPU, rendering em tempo real, ondas oceânicas, água, ondas que se quebram, fenômenos naturais, programação de vértice e fragmento, shader.

Abstract

Modeling natural phenomena is one of the most challenging tasks in computer graphic. The ocean wave simulation is included in this challenge. With the GPU graphical processing, it is possible to use advanced rendering techniques with great realism in real-time applications. This work initiates the research field in modeling and rendering of natural phenomena at Computing Institute of UFF in the area of Visual Computation and Interface. This dissertation simulates the ocean wave behavior processing all geometric computation and rendering in GPU. The shape is defined by Gerstner´s equation where the water particles movement is simulated. It is possible to consider the representation as in deep water as in shallow water. The method also simulates the breaking waves near the shore and the topology deep sea influence in it called wave refraction, changing its geometric properties adequately. The rendering implemented in this work uses combinations of advanced real-time rendering techniques from tangent-space reflective bump mapping and environment mapping to Fresnel reflection and HDR. As illustrated by examples of results, this technique creates a very realistic waves animation and adjusted scenario.

Keywords

GPU, real-time rendering, ocean waves, water, breaking waves, natural phenomena, vertex and fragment programming, shader

Sumário

Lista de figuras

1
Introdução

Elementos da natureza (paisagem, nuvens, água, etc) são muito mais complexos de serem modelados que formas matemáticas, equações ou objetos construídos pelo homem (torus, cubo, mesas, cadeiras, etc). Como fazem parte de cenas naturais, estão muito mais presentes na nossa vida do que os objetos sintéticos. Por isso nos últimos tempos, a simulação de fenômenos naturais com realismo visual tem sido objeto de muitas pesquisas dentro da computação gráfica (Clua, 1999). A natureza por si só já é complexa, mas hoje é possível gerar imagens impressionantes, graças a modelagem de fenômenos naturais utilizando alguns modelos físicos e estatísticos. Entre esses, pode-se citar a simulação de líquidos, fogo e gás (Fedkiw, Stam e Jensen, 2001)(Foster e Fedkiw, 2001).

No entanto, criar e renderizar águas é uma das tarefas mais onerosas da computação gráfica (Premoze e Ashikhmin, 2001). Uma renderização realística da água requer que a incidência da luz do sol e a iluminação do céu estejam corretas, que a modelagem da superfície da água esteja perfeita e que o transporte de luz no corpo da água e o seu movimento seja captado corretamente.

Até pouco tempo nos games, a água era renderizada como uma superfície plana e apenas aplicava-se uma textura de modo a assemelhar-se com a cor da água, não sendo consideradas as propriedades físicas que a tornam realmente realística.

A maioria dos trabalhos considera a modelagem geométrica das ondas através de duas formas de modelagem: física ou empírica. No modelo físico, adota-se a equação de Navier-Stokes que é baseada na dinâmica dos fluidos para representar o movimento da água através do tempo (Kass e Miller, 1990) (Foster e Metaxas, 1996) (Chen e Lobo, 1995). Devido a seu grande custo de processamento, este modelo ainda é inviável para processamento em tempo real. A abordagem com base totalmente física requerida para estudos científicos é bem diferente da abordagem para jogos em termos de precisão e fidelidade dos

cálculos. A outra forma de modelagem é através dos modelos empíricos, os mais conhecidos são de Fournier e Reeves (1986), Peachey (1986), Ts´o e Barsky (1987), Imamiya e Zhang (1995). Estes modelos são baseados no modelo clássico de ondas de Gerstner (Kinsman, 1965) em que a busca para a representação da fidelidade visual é maior do que a fidelidade aos fenômenos físicos.

O objetivo deste trabalho é modelar e renderizar a superfície das águas do oceano em tempo real na GPU representando inclusive efeitos complexos como a influência do vento, do relevo do fundo do mar e o efeito de ondas se quebrando. Para isso será desenvolvido um modelo de representação da geometria e um sistema de visualização de ondas, unindo diversos aspectos de *rendering*. Este sistema poderá ser usado em games, simuladores ou em aplicações que necessitem renderizar oceanos em tempo real.

O avanço na indústria de hardware possibilita que as imagens sintéticas atuais tenham um realismo em tempo real quase tão fiel quanto a renderização *offline*. Nesta dissertação será utilizado a GPU (*Graphic Process Unit*) para o processamento dos vértices e dos fragmentos. A GPU é um hardware dedicado a tarefas de computação gráfica e seu processamento e arquitetura em pipeline, torna possível o processamento de vértices e *pixeis* de forma extremamente rápida, liberando a CPU para outras tarefas no processamento em tempo real.

Para a definição da geometria da superfície oceânica, este trabalho seguirá a implementação de um modelo físico empírico. A equação da superfície trocóide é usada como base para a modelagem da forma da onda de acordo com o trabalho realizado por Fournier e Reeves (1986). A animação e a renderização é feita utilizando um modelo de sombreamento (*shader*) programado na linguagem Cg[1] da NVIDIA, sendo aplicadas técnicas de reflexão por bump mapping no espaço da tangente, environment mapping, reflexão de Fresnel (Wloka, 2002) e HDR (High Dynamic Range).

Esta dissertação está organizada em 9 capítulos. O **Capítulo 2** trata das pesquisas anteriores em torno deste assunto, desde as primeiras técnicas (que ainda são utilizadas) até as mais atuais para modelagem e animação de oceano.

---

1 *Linguagem de programação de shader (http://developer.nvidia.com/object/cg_toolkit.html)*

O **Capítulo 3** considera o conceito de ondas e como ele será incluído na abordagem desta dissertação. O **Capítulo 4** trata da GPU, seus conceitos básicos, evolução e aspectos que são utilizados na implementação desenvolvida. O **Capítulo 5** define a estrutura da implementação desenvolvida neste trabalho mostrando o sistema criado para a simulação. O **Capítulo 6** apresenta a modelagem geométrica da onda utilizada e implementada no sistema desenvolvido. O **Capítulo 7** apresenta as técnicas empregadas e implementadas para a renderização realística da superfície do oceano. O **Capítulo 8** descreve os resultados obtidos com a abordagem e a implementação utilizada. O **Capítulo 9** apresenta conclusões e possibilidades de trabalhos futuros.

2
Revisão da teoria: a evolução das técnicas de modelagem e animação de águas

Tem havido grande evolução das técnicas de computação gráfica na modelagem realística da água, bem como na sua renderização e animação (Iglesias, 2004).

Simular o movimento da água não é apenas modelar uma superfície plana, concentrar esforços na renderização, no efeito de reflexão e aplicar uma textura azul. A água por ser um fluído, pode mover-se em direções e caminhos complexos. A variável "tempo" deve ser sempre incluída nas equações para garantir o movimento do fluído e animação adequada de seu deslocamento.

O termo realismo possui diferentes significados na computação gráfica e depende exclusivamente do objetivo a que se deseja alcançar. Em contrapartida, um modelo de iluminação complexo de cena, não é geralmente, o esperado em visualização científica no entanto este é fundamental na geração de cenas em filmes gerados por computação gráfica.

Alguns trabalhos recentes abordam diferentes níveis de realismo na simulação de águas (realismo físico e foto-realismo) (Adabala e Manobar, 2002) (Ferwwerda, 1999)

O início dos anos 80 foi marcado como o ponto inicial da pesquisa em modelagem computacional e renderização de fenômenos naturais. Nesse período, as pesquisas se concentraram em representar uma grande massa de água sem bordas, como o oceano.

A primeira tentativa de renderizar as ondas da água utilizou a técnica de *bump mapping* (Blinn, 1978). Esta técnica também é usada neste trabalho (como mostra a seção 6.3). Este método permite obter superfícies com rugosidades realistas através da perturbação da normal da superfície modelada.

Outras referências importantes dos anos 80 foram dois conjuntos de slides apresentados na Siggraph. O primeiro deles é a "Pyramid" slide de Gary Demos

(Pyramid, 1981), onde as ondas do pôr-do-sol são obtidas por “bump mapping” de uma superfície lisa com formas de ondas cicloidais. O segundo slide é o “Nigth Castle”de Ned Green, na coleção de 1982, o qual usou ondas senoidais para a técnica de “bump mapping”.

Fishman e Schachter (1980) introduziram a técnica de “*height field*” e posteriormente Max (1981) foi outro método utilizado visando o realismo na simulação do oceano principalmente a visualização da textura do mar quando a câmera está próxima da superfície.

Mais tarde, Max (1981) usou uma abordagem diferente para renderizar a superfície das ondas para o seu famoso filme “Carla´s Island” como mostra a figura 1. Seu modelo hidrodinâmico simples foi baseado na idéia de que um modelo de onda é representado por uma solução aproximada (válido apenas para ondas de pequenas amplitudes) em que a velocidade da onda $\boldsymbol{v}$ é proporcional a raiz quadrada do comprimento de onda *L*.

$$v = \sqrt{\frac{gL}{2\pi}} = \sqrt{\frac{g}{k}}$$

onde

$\boldsymbol{k=2\pi/L}$ - é chamado o número da onda (o espacial análogo da freqüência);

$\boldsymbol{g}$ - representa a aceleração da gravidade.

Max também usou a primeira aproximação linear do modelo de Stokes (uma série de Fourier infinita a qual se assemelha as ondas trocoidas quando a série é usada com termos além das de terceira ordem).

O esquema de renderização utilizado em Max(1981) foi um modelo de “ray tracing” em que ondas do oceano e ilhas foram renderizadas por algoritmos diferentes mas relacionados entre si.

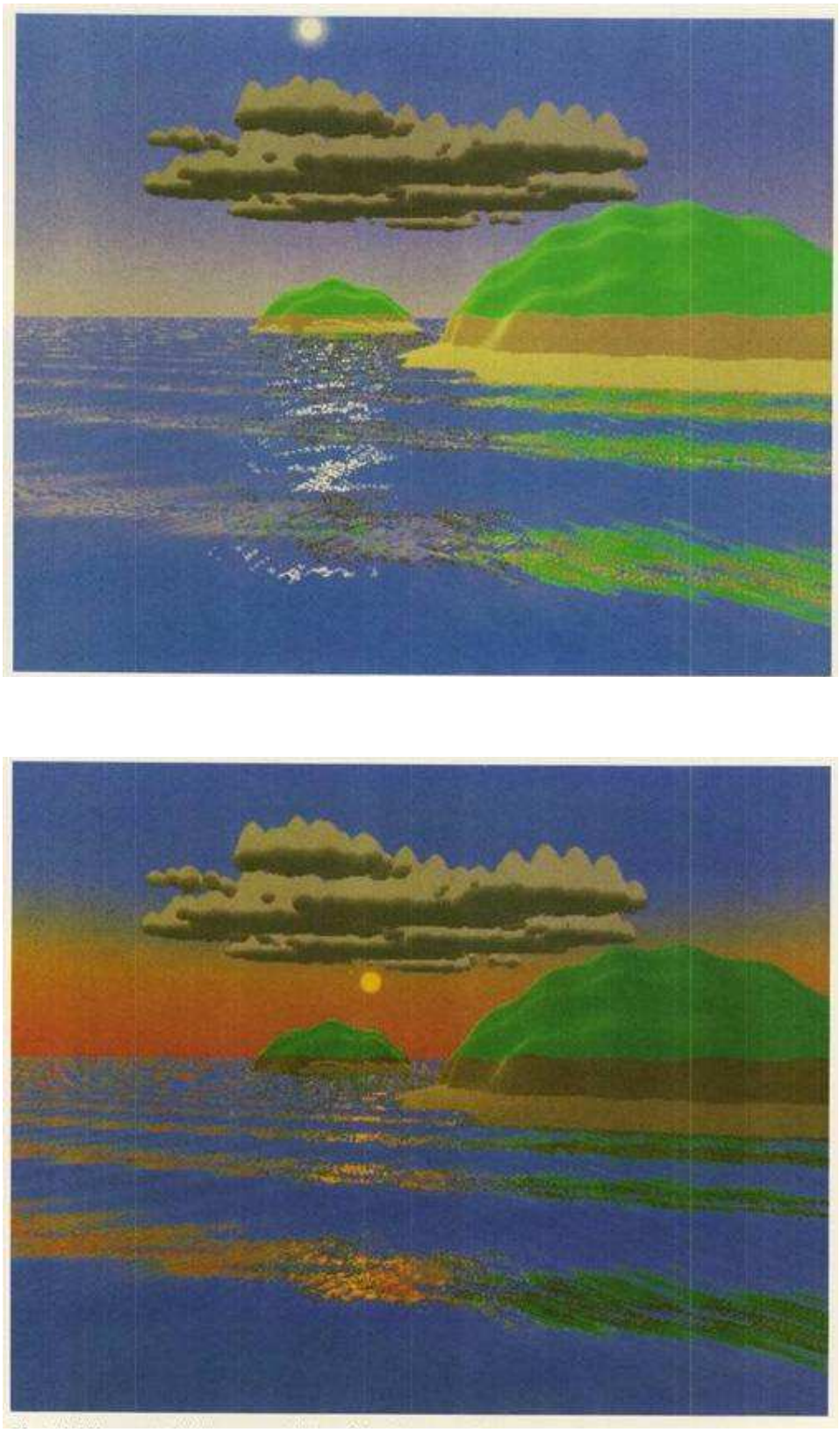


Figura 1 - Cenas do filme Carla´s Island (Fox e Waite, 1984)

Entre 1986 e 1988 com a preocupação da interação entre sólido e líquido, foram desenvolvidos técnicas que simulam refração e colisões. Dentre estes trabalhos, destaca-se o sistema de partículas de Reeves (1983) usados na simulação de espumas nas ondas. Peachey (1986) considerou que a superfície pudesse ser modelada usando "*height field*" e também conseguiu grande realismo em seu trabalho. Fournier e Reeves (1986) é um clássico sempre referenciado nos trabalhos atuais, baseado no modelo de Gerstner-Rankine proposto na oceanografia séculos atrás (Gerstner, 1809).

No modelo de Fournier-Reeves, os autores assumem que a partícula da água descreve uma órbita estacionária circular ou elíptica. Este é o modelo geométrico adotado neste trabalho (como mostra a seção 5.2). Formas de ondas realistas e outros efeitos necessários tais como os relacionados à profundidade (refração e quebras) e o vento, podem ser reproduzidos variando-se alguns parâmetros da equação da órbita deste modelo. Para controlar o formato do oceano Founier e Reeves (1986) introduziram alguns "trens de ondas", i.e. grupos de ondas compartilhando as mesmas características básicas (altura, período e amplitude de onda) e a mesma fase original. No trabalho Founier e Reeves (1986) também foram gerados efeitos de *spray* e espuma.

Até o final de 1988, alguns fenômenos físicos relacionados à água que ainda não havia sido explorados, começaram a ser considerados. A maior questão era a descrição exata da dinâmica de fluídos.

Por outro lado, alguns efeitos interessantes não haviam sido considerados até então, tais como: simulação de ondas refletidas, a interação entre luz e água, a análise caótica, etc. Para superar essas limitações, vários novos modelos foram criados para simular a dinâmica de fluídos. A princípio, estes novos modelos podem ser agrupados em dois grupos de diferentes abordagens:

1 - Dinâmica de fluídos baseada na interação de um número grande de partículas.

2 – Dinâmica de fluídos descrita pela solução de um conjunto de equações diferenciais parciais.

Os trabalhos relacionados à primeira abordagem que tentaram simular a dinâmica de fluídos através da interação com um grande número de partículas

foram os trabalhos de Goss (1990), Miller e Pearce (1989), Sims (1990) e Tonnesen (1991) onde os autores estudaram as forças de atração e repulsão entre as partículas para simular vários graus de viscosidade do fluído e o estado da mudança da matéria tal como o derretimento.

A segunda abordagem procura resolver diretamente um sistema de equações diferenciais parciais (PDE) descrevendo a dinâmica de fluídos (Kass e Miller, 1990). Esta técnica cria resultados muito realísticos em termos de simulação física. O problema principal é que uma simulação muito refinada ou exata da dinâmica de fluídos requer um cálculo computacional do movimento dentro de um determinado volume de controle. Isto torna o tempo de computação para cada iteração no mínimo proporcional a resolução ao cubo, tornando a computação exageradamente cara. Seguindo esta abordagem de pesquisa, os trabalhos desenvolvidos foram simplificando estes cálculos tornando sua utilização viável computacionalmente.

Um modelo bastante realístico pôde ser obtido usando a equação de Navier-Stokes, a mais detalhada de todos os modelos de fluídos. Devido a esta característica e algumas simplificações dos termos, a equação de Navier-Stokes tem sido amplamente utilizada na computação gráfica para simulação do movimento da água (Chen e Lobo, 1995) (Chen, Lobo, Hughes e Moshell, 1997) (Foster e Metaxas, 1996) (Foster e Metaxas, 1997a) (Foster e Metaxas, 1997b) (Foster e Metaxas, 2000) (Witting, 1999).

O que se tem visto é que o movimento da água é um fenômeno bastante complexo e variável, incluindo efeitos difíceis de serem analisados. A descrição exata destes efeitos é normalmente caracterizada pela área da física denominada dinâmica de fluídos ou fluido-dinâmica computacional (CFD). Recentemente vêm se destacando nesta área pelo seu grande realismo e aplicações em efeitos especiais de cinema os trabalhos de Fedkiw, Stam e Jensen (2001), Geiger, Leo, Ramussen, Losasso e Fedkiw (2006), Losasso, Fedkiw e Osher (2006) e Irving, Guendelman, Losasso e Fedkiw (2006).

Finalmente, seguindo a linha de Fournier e Reeves (1986) e Tessendorf (1999), pode-se citar o trabalho de Gonzato e Le Saec (1997, 1999 e 2000) que também tentam simplificar a física e se preocupar com a visualização realística da cena.

3
Introdução a oceanografia

Ondas são deformações periódicas de uma interface (Tomczak, 2002). A superfície das ondas, em oceanografia, são consideradas deformações na superfície do mar, devido à deformações na interface atmosfera-oceano. As deformações se propagam com a velocidade da onda enquanto as partículas descrevem movimentos orbitais ou oscilatórios e normalmente se mantém na mesma posição.

A superfície de onda ideal (figura 2) é senoidal: com a crista e a calha da onda tendo formas idênticas e a onda tendo um comprimento de onda fixo. Possui uma órbita progressiva com as partículas de água abaixo da onda se movendo com uma trajetória orbital que realiza um ciclo completo com a passagem de uma onda completa.

A distância (normalmente medido em metros) horizontal entre duas cristas (ou calhas) adjacentes é definida como o **comprimento de onda (L)**. A distância vertical do topo de uma crista até o fundo da calha adjacente é definida como a **altura da onda (H)**. A distância vertical da metade da altura da onda até a superfície do fundo do mar é definida como a **profundidade da água (h)**. Estes elementos sãos mostrados na figura 2.

Em uma escala temporal (normalmente medido em segundos), o intervalo de tempo entre a passagem de duas cristas consecutivas sobre um mesmo ponto fixo é definido como o **período da onda (T).** O inverso do período, que mede o número de ciclos de onda completos dentro de um intervalo unitário de tempo, é denominado **freqüência da onda (f)**. A velocidade com que a crista da onda move-se horizontalmente através da superfície do oceano é definido como a **celeridade da onda(c)** ou **velocidade da fase da onda** e normalmente é medido em metros por segundo (não descrito na figura).

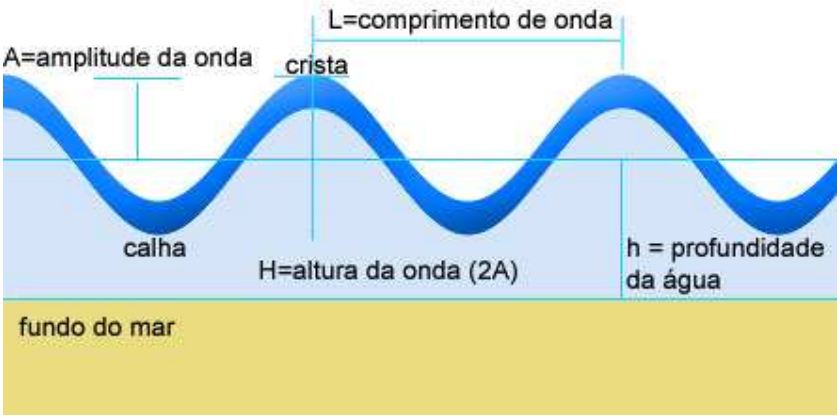


Figura 2 - Onda harmônica ou senoidal

Os principais elementos podem ser representados por:

- período **T**
- freqüência **f = 1/T**
- freqüência angular **ω = 2π/T**
- comprimento de onda **L**
- profundidade da água **h**
- celeridade da onda **c = L/T**
- altura da onda **H = 2A** (A = amplitude)
- *steepness* (agudez da onda) da onda **p = H/L**

3.1. Classificação da onda

As ondas podem ser classificadas de acordo com 4 aspectos:

- Profundidade da água;
- Método de geração;
- Período das ondas;
- Relacionamento com forças gerativas.

Os detalhes das características de cada uma destas formas de classificação são descritos a seguir.

3.1.1.Profundidade da água

Geralmente, a celeridade da onda (c) é proporcional ao comprimento de onda (L) e ao período (T), sendo também influenciado pela profundidade da água (h).

Em águas profundas, o deslocamento das partículas forma círculos. Em águas rasas, a partícula se movimenta de acordo com um traçado que se aproxima de uma elipse achatada. A mudança da forma das ondas, de água profunda para rasa, é observada quando o comprimento de onda L torna-se duas vezes maior que a profundidade da água *h*. Outra mudança na propriedade da onda ocorre também quando L = 20*h* quando ocorre o aumento da amplitude e diminuição do comprimento da onda até quebra-se na costa. A figura 3 esquematiza esta influência.

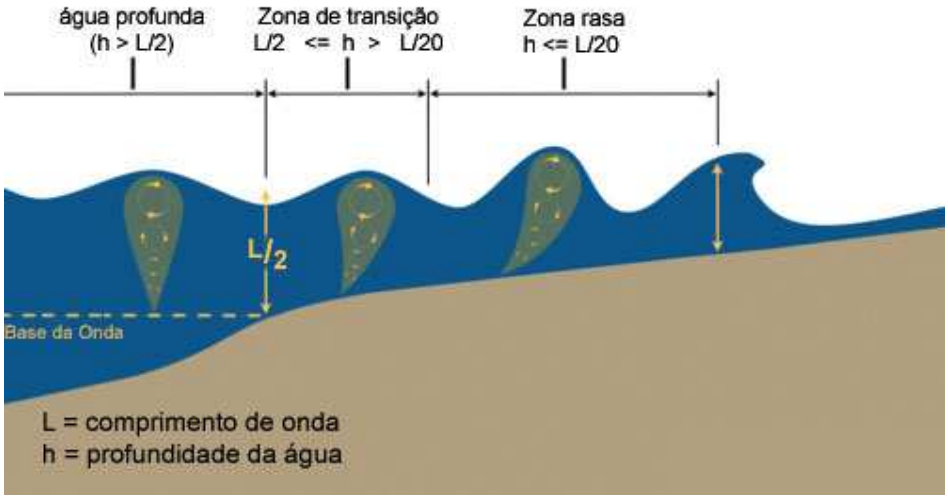


Figura 3 – Transição e variação das ondas

3.1.1.1. Ondas de águas profundas

Quando h é maior ou igual a metade do comprimento de onda, as ondas oceânicas não são afetadas pela profundidade da água. O diâmetro da órbita das partículas de água nestas águas profundas diminui à medida que a profundidade aumenta abaixo da superfície, chegando a zero quando h=1/2 L (figura 4).

Com isto o comprimento de onda varia uma vez que a celeridade da onda(c) é diretamente proporcional ao comprimento de onda. Isto significa que ondas com grandes comprimentos de onda possuem uma velocidade maior na superfície do oceano.

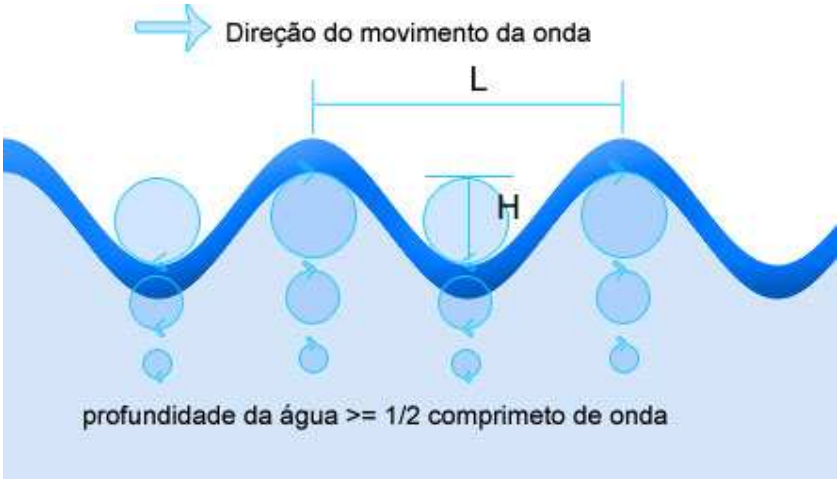


Figura 4 – Ondas de águas profundas

3.1.1.2. Ondas de águas rasas

Quando h é menor ou igual a 1/20 do comprimento de onda, as partículas de água das ondas oceânicas mudam sua forma de acordo com a profundidade da água. As órbitas das partículas de água passam a ter a forma de elipses alongadas com o eixo maior na direção horizontal e o eixo menor da direção vertical. Somente o eixo menor decrementa na medida em que a profundidade abaixo da superfície aumenta. Deste modo, próximo ao fundo, no limite do movimento a trajetória das partículas da água é somente horizontal. Ela se move para frente e para trás com a passagem da onda (figura 5).

Assim a celeridade da onda é diretamente proporcional a profundidade da água. Isto significa que a medida que a profundidade da água diminui , a velocidade da onda também diminui.

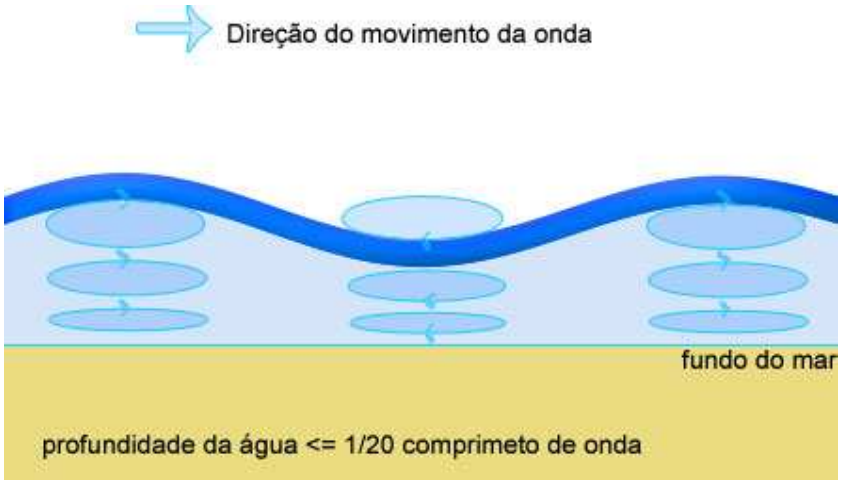


Figura 5 – Ondas de águas rasas

3.1.1.3. Grupos de ondas

A superposição de duas ondas com freqüências angulares quase iguais ω1 e ω2 produz grupos de onda ou pacotes (figura 6). Individualmente, as cristas das ondas viajam com velocidade de fase (idêntica à celeridade da onda) c, pacotes de onda viajam com velocidade do grupo.

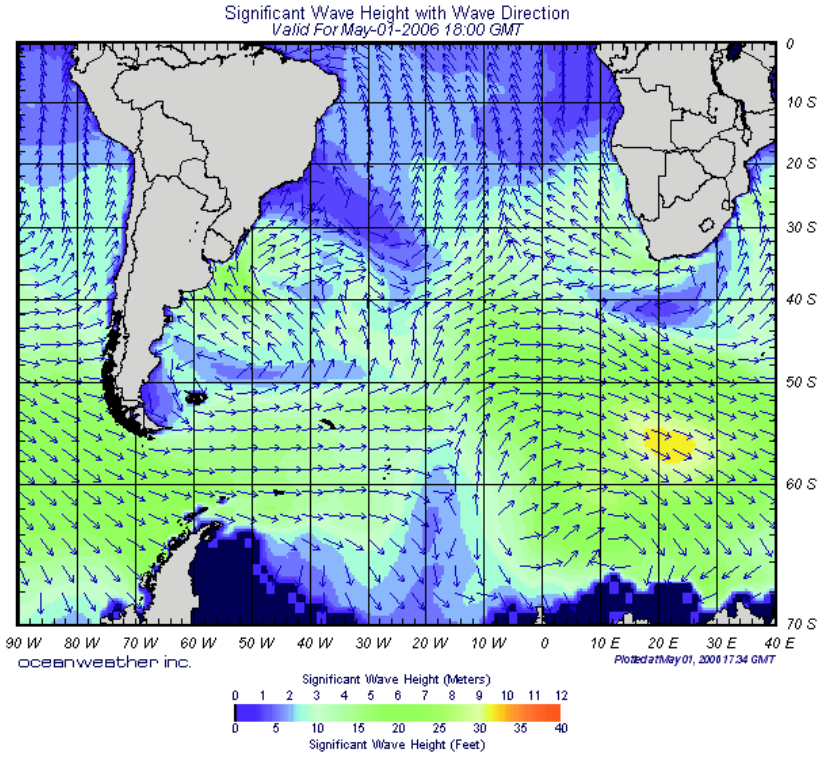


Figura 6 – Grupos de onda.

3.1.2.Método de geração

As ondas podem ser classificadas pelo método de sua geração. **Ondas de vento** são geradas quando o vento entra em contato com a superfície da água e uma quantidade de movimento é transferido do vento para a água. **Ondas de impacto** (como por exemplo os Tsunamis) podem ser geradas na superfície da água por terremotos ou qualquer outra forma de impacto (mesmo em pequena escala, como uma pedra jogada em uma lagoa).

As ondas provocadas pelo vento formam ondas direcionais, ou seja, as ondas se movem na direção da força do vento. As ondas provocadas por impacto, como os Tsunamis, formam ondas circulares, ou seja, as ondas se propagam na direção radial ao ponto de impacto.

3.1.3.Período das ondas

As ondas também podem ser classificadas pelo período de onda. A energia cumulativa é distribuída nas ondas oceânicas, onde o princípio de geração de forças e o princípio de restauração de forças muda quando o período das ondas aumenta.

As menores ondas (**ondas capilares**) possuem período pequeno menor que 0,1 segundos, são geradas por um sopro de vento e por serem tão pequenas são restauradas pela tensão superficial (Garrison, 2004). A onda mais comum (**ondas de gravidade**) com período entre 1 e 30 segundos são geradas pelo vento e tempestades, sendo restauradas pela força da gravidade. Ondas com período maior que 5 minutos (**ondas longas**) são geradas por tempestades intensas e por terremotos, sendo restauradas pela força da gravidade e pela força de Coriolis (Wiki, 2006). As **marés** são ondas de período entre 12 e 24 horas, são geradas pelo sol e pela lua e restauradas pela fricção com o fundo do mar e pela força de Coriolis.

3.1.4.Relacionamento com forças gerativas

Ondas que correm independentes da continuidade de sua força geradora (como as ondas de impacto) são chamadas **ondas livres**. Ondas que são

dependentes de sua força geradora para a continuidade de sua existência (como as marés) são chamadas **ondas forçadas**. Algumas ondas de vento que estão iniciando sua geração (como uma intensa tempestade) podem ser classificadas como **ondas livres/forçadas**.

3.2. Arrebentação

Em águas profundas, somente o comprimento de onda e o período afeta a velocidade da onda. A medida que a onda aproxima-se das águas rasas, ou águas em que a profundidade é menor que a metade do comprimento de onda, o fundo do oceano começa a influenciar o formato e a velocidade da onda. A altura da onda aumenta, e a crista começa a ficar mais pontuda.

As ondas podem-se quebrar quando a razão entre a altura e o comprimento da onda (agudez da onda) ultrapassar 1/7 ou quando a crista da onda aproximar-se de um ângulo de 120º (Kinsman, 1965).

As ondas que se quebram (ou arrebentação) podem ser descritas em três diferentes tipos: quebras surgindo (vagalhão), quebras mergulhando (em espiral) e quebras em derrame. É possível observar exemplos destas ondas na praia de acordo com a inclinação do fundo como mostra a figura 7.

Ondas que se derramam ocorrem em praias com pequena inclinação. Estas ondas se quebram lentamente a partir da crista, continuando o processo por longas distâncias enquanto se aproximam da praia.

Ondas que se quebram mergulhando (em espiral) acontecem em praias onde a inclinação é moderadamente íngreme. Este tipo de onda normalmente cria um tubo até que a onda quebre. Os surfistas profissionais adoram esse tipo de onda.

Quebras surgindo (vagalhão) acontecem em praias onde a inclinação é muito íngreme. A onda não quebra exatamente. Ao invés disso ela rola sobre a inclinação da praia.

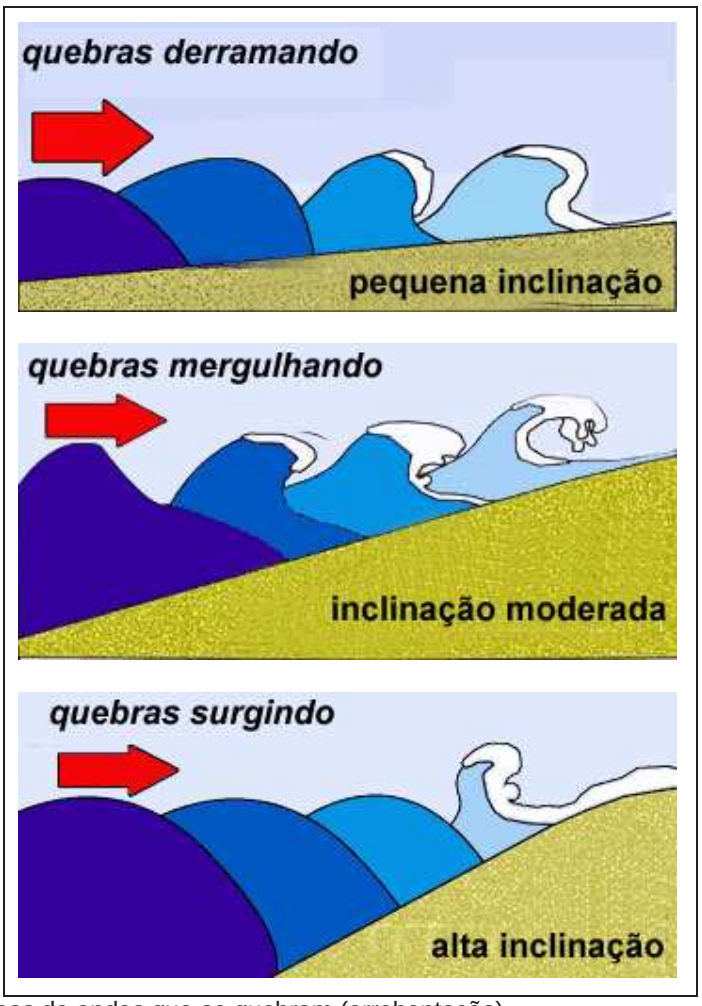


Figura 7 – Tipos de ondas que se quebram (arrebentação)

4
Programação em GPU

Os processadores gráficos (GPU) produzidos atualmente estão oferecendo grande poder computacional, flexibilidade de processamento e precisão. As linguagens de alto-nível vêm ganhando espaço dentro do hardware gráfico tornando utilizável este poder computacional.

O objetivo deste capítulo é justamente mostrar os processos, vantagens e aplicações da programação em GPU utilizando linguagens de alto-nível.

4.1.
Linguagens para o hardware gráfico

As placas gráficas deixaram de ser apenas uma interface de hardware para enviar dados ao monitor e passaram a ser muito mais que isso. Ganharam memória e um processador tão poderoso que podem ser utilizadas não apenas pelos profissionais de computação gráfica, como também de outras áreas da ciência da computação devido a sua arquitetura, processamento em pipeline e a flexibilidade de programação na própria GPU.

No passado, para se programar na GPU, a única opção era escrever códigos em Assembly. Com a evolução do hardware gráfico, para utilizar de toda sua capacidade, programar em Assembly ficou praticamente inviável. Com a necessidade de se programar em alto-nível, surgiram linguagens de programação. Baseadas no RenderMan, desenvolvido pela Pixar em 1998, mas com a sintaxe parecida com a linguagem C, as três linguagens mais difundidas de alto-nível são: **Cg** (C for Graphics), **HLSL** (High Level Shader Language) e **GLSL** (OpenGL Shader Language).

O Renderman, apesar de poderoso, é uma linguagem de processamento offline, ao passo que as linguagens de alto-nível Cg, HLSL e GLSL são destinadas a aplicações em tempo real e executam diretamente na GPU, alterando

a forma convencional de tratamento de vértice e fragmentos no processo de renderização.

4.2. Vértices e fragmentos

Para entender o contexto de uma linguagem para GPU é necessário entender como as GPUs realizam o *render* das imagens. Para isso, é importante conhecer a evolução dos hardwares gráficos e então saber como explorar o pipeline de render moderno do hardware gráfico.

4.2.1. Evolução das GPUs

A cada ano, a indústria de hardware gráfico avança com uma enorme velocidade. Existem três forças que direcionam para este crescimento como mostrado na figura 8. Primeiro, a indústria de semicondutores com a constante miniaturização: os transistores ocupando menos espaço nos circuitos (historicamente conhecido como Lei de Moore). Também o hardware gráfico cada vez mais barato, rápido e com novas funcionalidades. A segunda força é a quantidade enorme de computação requeria para simular o mundo, ou seja, permitir reproduzir o espaço 3D com cada vez mais realismo. A terceira força, o processamento paralelo na geração de imagens é o desejo constante dos humanos de serem estimulados e entretidos visualmente. Talvez esta seja a maior razão da necessidade crescente de evolução das placas gráficas, haja vista o público de games que vêm crescendo a cada dia, maravilhados com a sua qualidade e o seu realismo.

Figura 8 - Forças que direcionam as inovações do hardware gráfico (Fernando e Kilgard, 2003)

4.2.2. Pipeline gráfico

Um pipeline é uma seqüência de estágios que operam em paralelo e em uma ordem fixa (Randima e Kilgard, 2003). Cada estágio recebe a entrada do estágio anterior e envia sua saída para o próximo estágio. Como em uma linha de montagem de automóveis, em que vários automóveis são produzidos ao mesmo tempo, com cada automóvel em diferentes estágios da linha de produção, um hardware que usa o pipeline gráfico convencional processa vários vértices, primitivas geométricas, e fragmentos na forma mostrada na figura 9.

A aplicação 3D envia à GPU uma seqüência de vértices na forma de primitivas geométricas: normalmente polígonos, linhas e pontos. Cada vértice tem uma posição (coordenada x, y e z), mas também possui outros atributos tais como uma cor principal, uma cor secundária (ou *specular*), uma ou múltiplas coordenadas de texturas, e um vetor normal. O vetor normal (embora seja uma característica pontual) indica qual é a direção da superfície a que pertence a face do vértice, e é normalmente usado nos cálculos de sombreamento e iluminação.

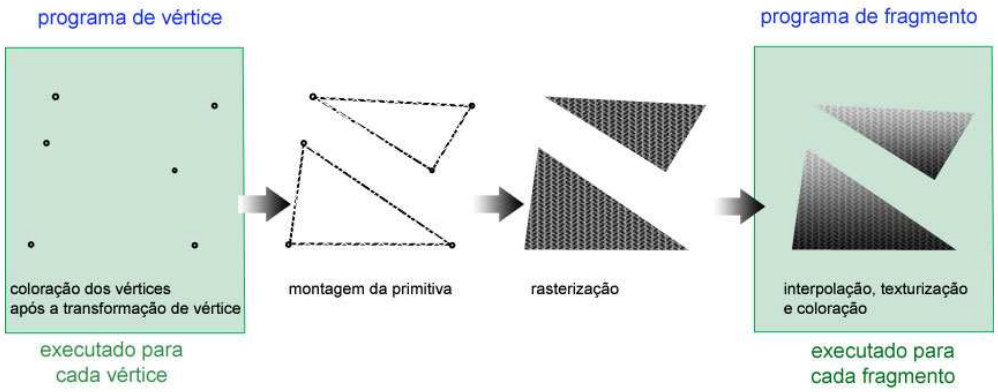


Figura 9 - Pipeline gráfico

4.2.3.
Pipeline gráfico programável

A tendência dominante dos designer de hardware gráfico é o esforço em tornar a GPU cada vez mais programável. De acordo com a figura 10, pode-se observar dois estágios (processamento de vértice e processamento de fragmentos) quebrados em unidades programáveis. São nestas duas unidades que é possível programar usando linguagens de alto nível como Cg, HLSL e GLSL.

O processador de vértice programável é a unidade de hardware onde são executados os programas de vértice (vertex shaders), e o processador de fragmentos programável é a unidade onde rodam os programas de fragmentos (fragment shaders).

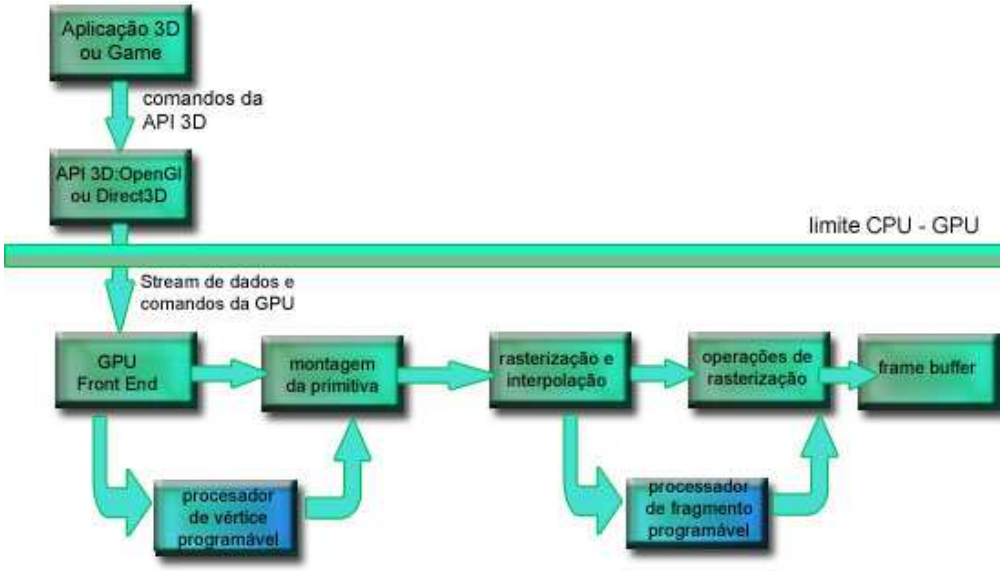


Figura 10 - Pipeline gráfico programável

4.3. As diferentes linguagens de alto nível para a programação em GPU

Como visto no item 4.1 programar em assembly é inviável de acordo com o grande número de instruções e complexidade dos hardwares gráficos. Para resolver este problema, tornando a leitura do código mais próximo da linguagem humana e consequentemente mais reutilização e portabilidade, surgiram as três linguagens de alto-nível: Cg, HLSL, GLSL.

4.3.1.Cg

Criada em 2003 (NVIDIA, 2003), NVIDIA e Microsoft colaboraram para o desenvolvimento da linguagem que a NVIDIA denominou “Cg” que significa “C for Graphics” devido a sua sintaxe semelhante a linguagem C criada em 1970 nos laboratórios Bell. Cg é compatível com OpenGL e Direct3D e roda na plataforma Windows, Linux, Mac OS X e em console de games.

4.3.2.HLSL

HLSL (High Level Shader Language) (MSDN, 2003) é praticamente idêntica a linguagem Cg da NVIDIA, nome dado pela Microsoft para diferenciá-la da Cg.

A diferença principal é que HLSL é compatível apenas com a API Direct3D e específico para a plataforma Windows.

4.3.3.GLSL

OpenGL Shading Language (Rost, 2004) foi criada em 2003 como uma extensão do OpenGL concebida pela organização ARB fundada pela Silicon Graphics. GLSL é compatível com a API OpenGL e as plataformas Windows, Mac e Unix.

4.4. Criando efeitos com a linguagem Cg

Como citado anteriormente, existem linguagens de alto-nível que facilitam a programação de shaders criando programas complexos de pixel e vértice. A linguagem Cg será utilizada neste trabalho.

Através da utilização desta linguagem também é possível criar “efeitos”. Para criá-los, é necessário definir o formato do arquivo como sendo CgFx. Este formato de arquivo é análogo ao arquivo .fx da Microsoft utilizado no DirectX 9. Este formato permite que os *shaders* sejam definidos com parâmetros editáveis, por exemplo, pode-se aplicar o mesmo *shader* em dois objetos diferentes e seu comportamento será dependente do parâmetro definido em cada objeto separadamente.

A programação do sombreamento (*shader*) num formato de efeito tem importância principalmente para os artistas, pois ele poderá ver e ajustar o efeito na forma como ele realmente irá aparecer quando aplicado em uma cena. O efeito também pode ser editado dentro de softwares proprietários como o 3D Max e o Maya.

4.5. Hardware shading vs. software shading

Normalmente, a renderização por software e a renderização por hardware é realizada com as mesmas técnicas. Porém, a renderização por software se preocupa mais com a qualidade e flexibilidade do que com a velocidade. Já na

renderização por hardware, o tempo de renderização e a eliminação de gargalos são mais importantes que todo o resto. Pois não existe uma cena 3D em tempo real boa se ela não estiver realmente em tempo real (normalmente em 60 frames por segundo) (NVIDIA, 2002).

Existem grandes diferenças entre os shaders offline e os shaders em tempo real em função de sua concepção. Alguns artistas, por exemplo, aplicam um pequeno número de shaders em sua cena variando seus parâmetros de milhares de formas diferentes. Este é um exemplo típico da flexibilidade característica de um modelo de shader offline. Em contrapartida, ao se utilizar um shader que possibilita utilizar a técnica de bump-mapping, mas não é desejada no momento da cena, deve-se retirar esta funcionalidade do shader para ganhar em desempenho de renderização. A utilização do “efeito” permite este tipo de flexibilidade, mesmo em cenas de tempo real.

5
Estágios do sistema de simulação

O objetivo deste trabalho é de gerar imagens sintéticas e convincentes de ondas do oceano como vistas na praia ou em alto mar. Utilizaremos a programação na GPU criando um *shader* para animação e renderização da superfície das ondas do mar.

O sistema proposto para a simulação da onda no *shader* é composto de 4 estágios: geração da onda, modelagem da superfície, computação óptica e renderização da água. A figura 11 mostra estes estágios e suas conexões.

Os dois primeiros estágios serão realizados dentro da programação de vértice. A geração da geometria da onda será definida utilizando-se a equação de Gerstner (Tessendorf, 1999) com trajetória circular para simular a geometria em águas profundas e trajetória elíptica para simular a geometria em águas rasas.

Como a proposta do trabalho é a construção de um *shader* na forma de efeito (item 4.4), a modelagem da superfície será feita de uma forma muito simples. Basta gerar um objeto utilizando um software de modelagem e em seguida aplicar o *shader* desenvolvido neste trabalho sobre o objeto implementado. No caso das imagens aqui geradas, será considerado um retângulo no plano XZ. Esta é uma das grandes vantagens da solução proposta devido a sua reutilização e ganho em produtividade. Ou seja, a simplicidade da adequação deste efeito a um aplicativo 3D ou a um game. Modelando-se um simples retângulo no plano XZ, e aplicando este *shader* aqui implementado sobre o mesmo, este toma a forma de um oceano animado. Sobre este modelo, será gerado o *bump mapping* (item 6.3) para dar o efeito de rugosidade na geometria da superfície peculiar as pequeninas ondas que se formam no oceano.

Para os dois últimos estágios será desenvolvido um programa de fragmentos. A computação da óptica irá calcular a iluminação, reflexão de Fresnel

e HDR. No último estágio, unindo os três estágios anteriores e a contribuição da cor da água, finalmente será renderizado o oceano.

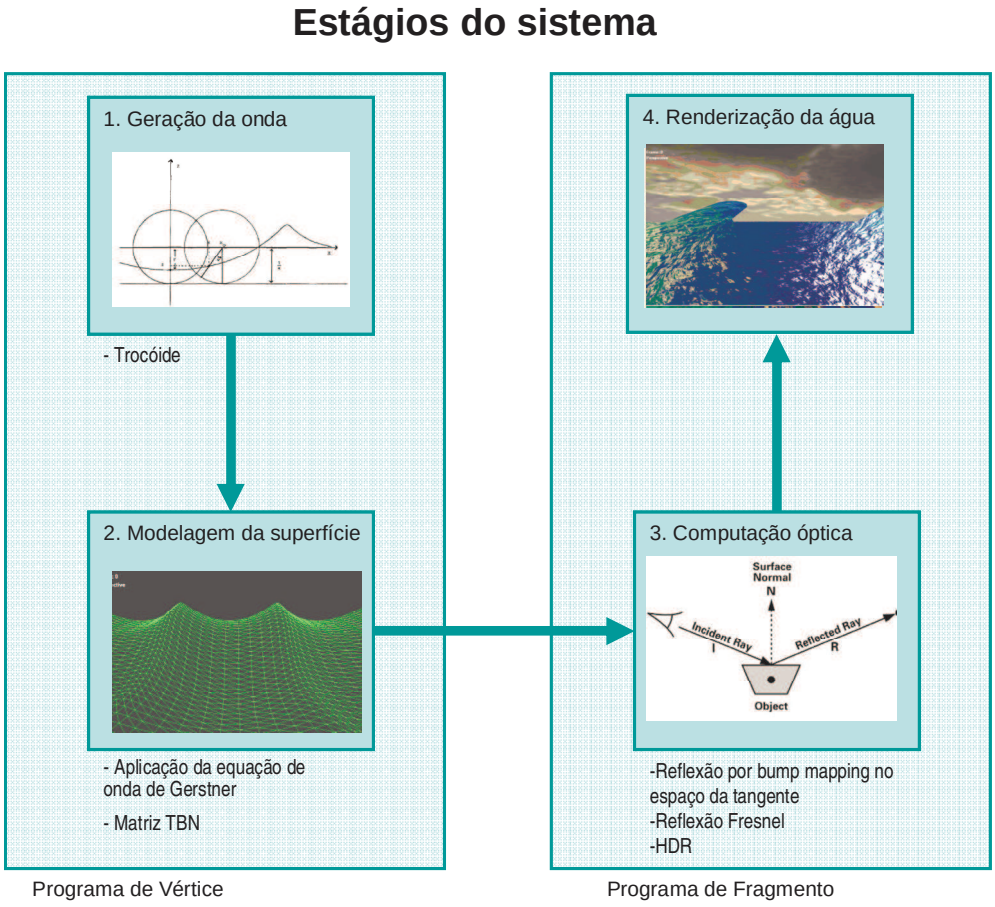


Figura 11 – Estágios do sistema de simulação

5.1.
O Modelo básico da onda

Usa-se aqui o modelo apresentado por Fournier e Reeves (1986). O movimento dos fluidos geralmente é descrito por duas formulações: Euleriana ou Lagrangiana. A formulação Euleriana é mais adequada na hidrodinâmica e ao estudo das ondas, especialmente para o desenvolvimento de modelos estocásticos na análise do mar. Considera-se um ponto (x, y, z) e tenta-se responder questões sobre as características do fluído neste ponto em função do tempo, como por exemplo, a velocidade:

$$U = f(x, y, z, t)$$

A formulação Lagrangiana, a princípio é mais apropriada para modelagem gráfica por tratar o oceano como sendo uma primitiva geométrica. Ela descreve a trajetória de um ponto (x_0, y_0, z_0) dado por uma posição de referência. Isto pode ser visto como a trajetória de uma partícula. Por exemplo, pode-se saber a velocidade no tempo t:

$V_x = f_x(x_0, y_0, z_0, t)$

$V_y = f_y(x_0, y_0, z_0, t)$

$V_z = f_z(x_0, y_0, z_0, t)$

5.2. Ondas de Gerstner

Para uma simulação efetiva do oceano, precisamos controlar a agudez (*steepness)* da onda. Como mostrado no capítulo 3, a onda perfeita tem a forma de uma senoide – como uma onda num lago calmo. Mas para simular o oceano, é necessário criar cristas com picos afinados e calhas arredondadas. Para representar esta onda com mais realismo será utilizado o modelo de onda de Gerstner. A equação da onda de Gerstner foi originada das bases da física muito antes do surgimento da computação gráfica (a mais de 200 anos atrás) como uma solução aproximada para uma equação de dinâmica de fluídos (Tessendorf, 1999). O modelo físico descreve a superfície em termos de movimentos de pontos individuais na superfície.

Será considerada que uma partícula descreve um movimento circular a partir de sua posição de repouso. O plano XZ é o plano do mar em repouso e Y representa a coordenada de altura ao plano da superfície do mar. Considerando o movimento no plano XY, a equação de Gerstner simplificada será o sistema:

$$\begin{aligned} x &= x_0 + r\sin(kx_0 - \omega t) \\ y &= y_0 - r\cos(kx_0 - \omega t) \end{aligned} \qquad (1)$$

Onde:

H = 2r é a altura da onda;

k = 2π/L é o número de onda;

L = 2π/k = gT^2/2π é o comprimento de onda;

T = 2π/ω é o período;

c = L/T = ω/k é a celeridade da onda (velocidade da fase) ou seja, a velocidade de viagem da crista como será visto no item 6.2.

Olhando a equação 1 como uma equação paramétrica percebe-se que se trata de uma trocóide, uma generalização da ciclóide. Esta equação representa a curva descrita por um ponto **P** que tem distância **r** do centro de um círculo de raio **1/k** que se move rolando sobre uma linha de distância **1/k** sob o eixo X (figura 12). Portanto, amplitude **A= r**, $\mathbf{x_0}$ e $\mathbf{y_0}$ são as coordenadas iniciais do mar em repouso, **ω** é a frequência, **t** o tempo e **k** o número da onda.

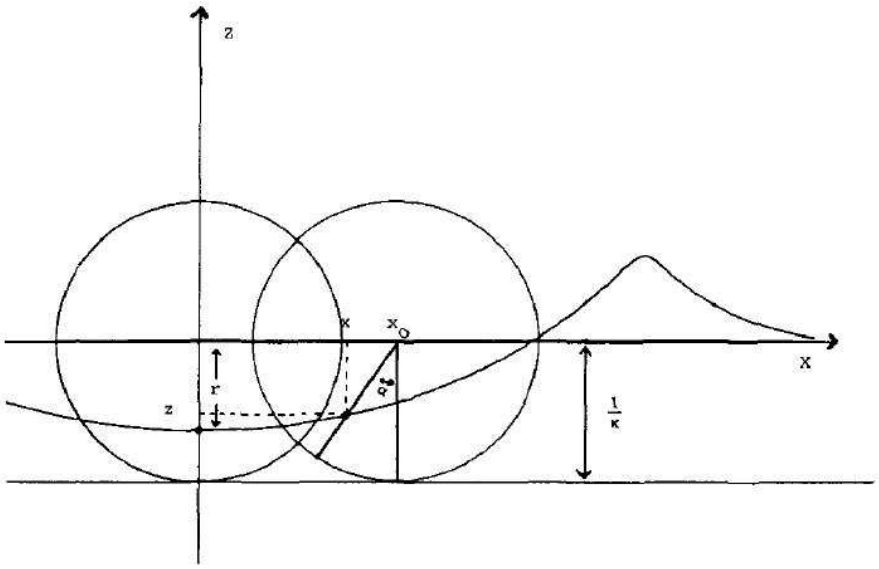


Figura 12 – A trocóide

Assumindo **Φ = kx0- ωt** como sendo a fase da onda, pode-se reescrever a equação como:

$$x = x_0 + r\sin(\Phi)$$
$$y = y_0 - r\cos(\Phi) \qquad (2)$$

Com este modelo básico é possível chegar as formas desejadas para uma cena realista. Por exemplo, alterando-se **kr**, obtem-se várias formas de onda como mostrado na figura 13.

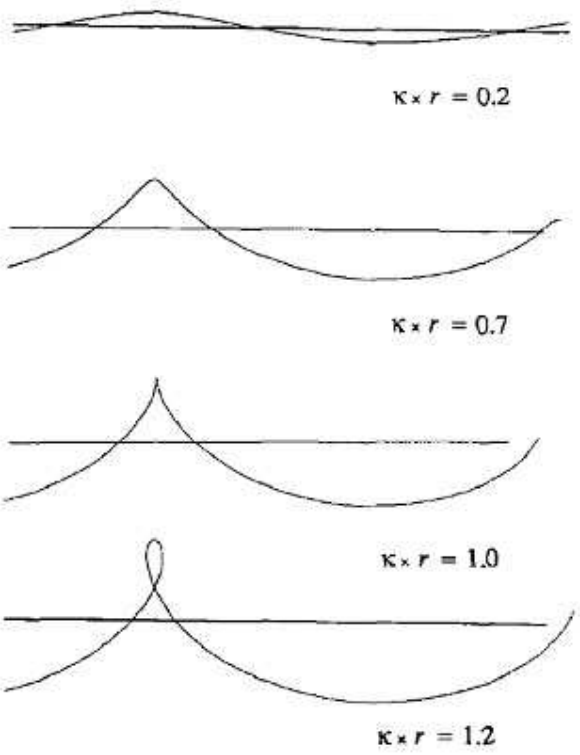


Figura 13 – Formas de onda variando o parâmetro *kr.*

5.3. Consideração dos estágios do sistema

A partir deste modelo básico, será definida a forma da onda e modelada e sombreada de acordo com a definição no capítulo 6 e 7. Desta forma, foi criado um *shader* com um programa de vértice e um programa de fragmento.

O programa de vértice definido do capítulo 6 implementa a equação de onda de Gerstner com variações dos atributos para simular a influência do fundo na formação da onda como a refração, influência do vento e quebra. O programa de vértice também calcula a matriz para o cálculo da iluminação.

O programa de fragmento definido no capítulo 7 utiliza das informações recebidas do programa de vértice para calcular a iluminação e definição das cores finais.

6
Geração da onda e modelagem da superfície

Como descrito no capítulo 4, esta dissertação utiliza a programação em GPU para implementar a simulação de onda.

Será descrito aqui cada passo até chegar a forma de onda final desejada.

Como o movimento de cada partícula da água é definido por um traçado circular (figura 14), com o emprego do *shader* é possível aplicar a equação 2 em cada vértice da figura que define a superfície do mar.

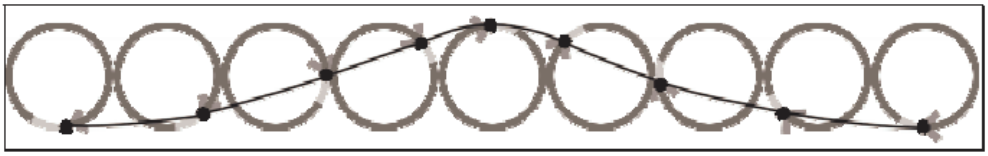


Figura 14 – Posição de uma partícula de água com o movimento da onda

Utilizando a linguagem Cg, este cálculo será efetuado no programa de vértice. Basicamente, nesta etapa, o programa de vértice executa os seguintes passos:

1. Receber como entrada um polígono retangular representando a superfície do oceano no plano XZ;
2. Transformar os vértices do polígono utilizando a equação 2;
3. Transformar as coordenadas atuais do espaço do objeto projetado;
4. Enviar dados do vértice transformado para o processador de fragmentos.

Desta forma, obtém-se o resultado mostrado na figura 15 exibida em *wireframe* (linhas que ligam os vértices definindo a estrutura da malha triangular).

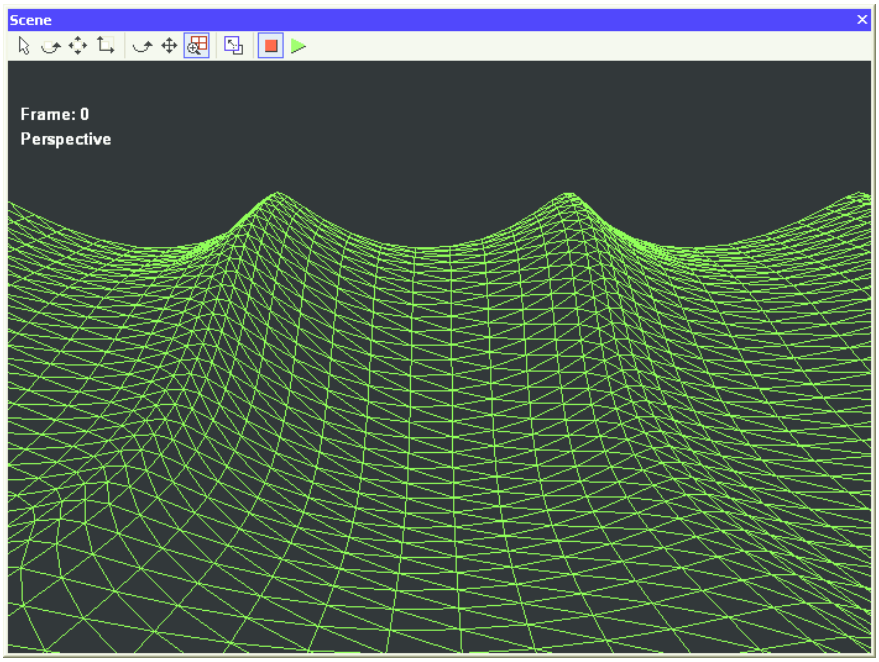


Figura 15 – Resultado com a aplicação da equação de Gerstner.

6.1. Acrescentando a influência do vento

Uma vez definido a forma básica da onda será acrescentado alguns efeitos especiais. Um deles é o efeito do vento sobre as ondas fazendo com que elas sofram uma inclinação na parte superior da onda em direção na mesma direção do vento.

Para acrescentar tal efeito, será adicionado mais um controle no ângulo de fase da equação de Gerstner, ou seja:

$\Phi = kx_0 - \omega t - \lambda \Delta y \Delta t$, será a equação da nova fase.

Onde:

λ é uma constante de proporcionalidade do vento

Pela expressão é possível constatar que a partícula será mais acelerada no topo e mais desacelerada na base da onda gerando uma projeção na forma da onda em direção a frente de onda (figura 16).

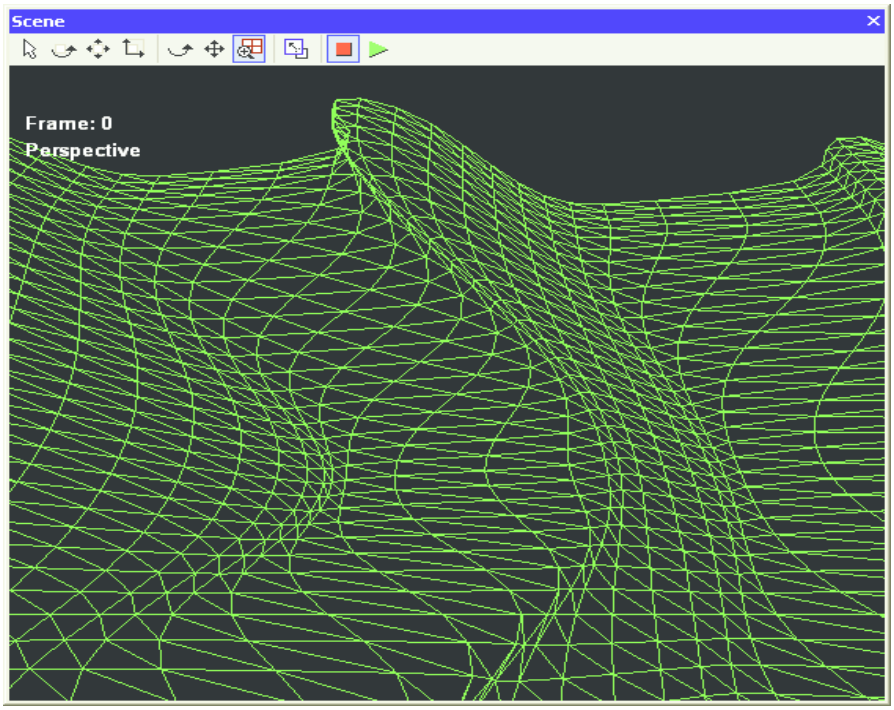


Figura 16 – Acrescentando a influência do vento.

6.2.
Refração e influência do fundo do mar sobre as ondas

Quando as ondas se propagam de águas profundas para águas rasas (quando está se aproximando da costa, por exemplo), quase todas as características da onda mudam assim que ela começa a sofrer a influência do fundo. Somente o período mantém-se constante. A velocidade da onda diminui com a diminuição da profundidade.

Assim que as ondas passam a sentir o fundo, um fenômeno chamado refração pode ocorrer. Quando as ondas entram na zona de transição (profunda para rasa), a parte da onda em águas profundas move-se mais rapidamente que a onda em águas rasas. Como mostrado na figura 17, essa diminuição na velocidade da fase da onda pode ser percebido numa visão aérea do mar. A medida que a onda sente o fundo do mar e sua velocidade e seu comprimento de onda diminui, a crista da onda tende a se alinhar com a costa marítima. De acordo com a equação:

$$L = cT$$

Onde:

- **L** é o comprimento de onda;

- **c** a celeridade da onda;

- **T** o período.

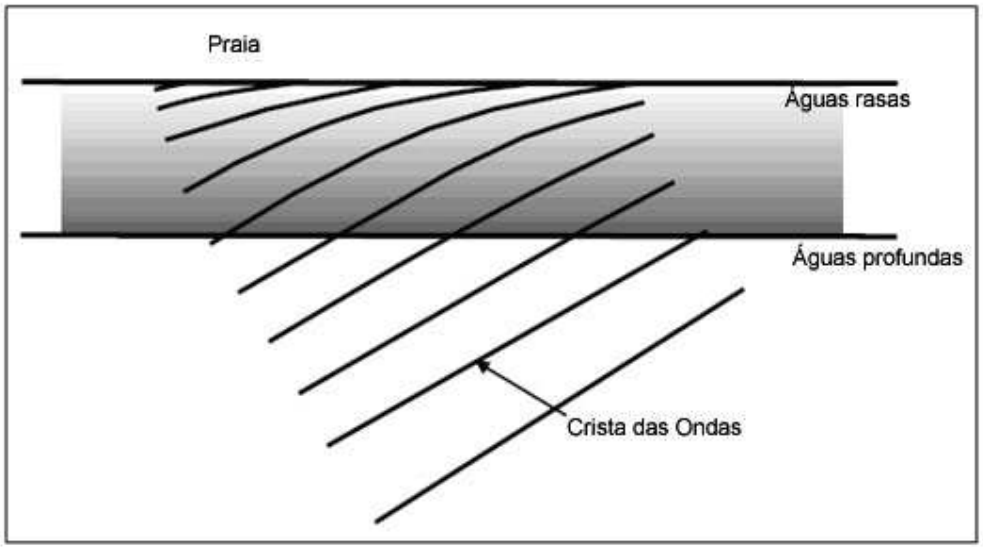


Figura 17 – Refração da onda ao se aproximar da costa (águas rasas).

Na zona de transição entre águas profundas e águas rasas, a celeridade (c) da onda (m/s) é calculada pela equação:

$$c^2 = \frac{g}{k}\tanh(kh) \qquad (3)$$

Onde:

- **k = 2π/L** é o número de onda;

- **g** é a aceleração da gravidade;

- **h** a profundidade da água.

Em águas profundas, onde a profundidade da água é maior que a metade do comprimento de onda, **kh** é um valor muito grande de modo que **tanh(kh)** é aproximadamente igual a 1. Portando, a celeridade em m/s é pode ser escrito como:

$$c^2 = \frac{g}{k} = \frac{gL}{2\pi}$$

ou escrito com um produto de celeridades,

$$c * c = \frac{gL}{2\pi}$$

e substituindo c = L/T, tem-se:

$$c * \frac{L}{T} = \frac{gL}{2\pi}$$

simplificando L,

$$c = \frac{gT}{2\pi}$$

ou seja, supondo **g = 9,8 m/s^2**,

$c = T * 1{,}56$ é a celeridade da onda em **águas profundas**.

Em águas rasas, onde a profundidade da água (h) é menor que 1/20 do comprimento de onda, **th** é um valor pequeno, logo **tanh(th)** é aproximadamente igual a **th**. Substituindo **tanh(th)** por **th** na equação 3 da celeridade e simplificando **k**, tem-se:

$$c^2 = \frac{g}{k}(kh) = gh$$

Extraindo a raiz quadrada, obtém-se:

$c = \sqrt{gh}$, como sendo a celeridade da onda em **águas rasas**.

6.2.1.O comprimento de onda de acordo com a profundidade

A diminuição da profundidade da água também altera o comprimento de onda, sendo que o período permanece constante (Kinsman, 1965). Se chamarmos de k_∞ o número de onda em uma profundidade infinita, uma boa aproximação para o número de onda k na profundidade h é:

$$k \tanh(kh) = k_\infty \qquad (4)$$

Quando **x** → **0**, então **tanh(x)** → **x**. Logo, em águas rasas, onde a profundidade é bem pequena, a relação fica:

$$k^2 h = k_\infty \text{ ou } k = \sqrt{\frac{k_\infty}{h}}$$

Quando **x** → ∞, então **tanh(x)** → **1**, logo **k** → **k**$_\infty$. Uma vez que **kh** = **2πh/L**, com uma relação **h/L** de 1/2 obtém-se um argumento de **π** para a tangente hiperbólica o que é praticamente igual a 1. Por esta razão, a "profundidade" tem relacionamento com o comprimento de onda e significa a relação **h/L** ser maior que 1/2 como citado anteriormente na figura 4. Uma boa aproximação para a equação 4 é:

$$k = \frac{k_\infty}{\sqrt{\tanh(k_\infty h)}}$$

6.2.2.A celeridade em relação à profundidade

Uma vez afetado o comprimento de onda, a celeridade da onda também é afetada como mostrado no item anterior, o que significa: **c/c**$_\infty$ = **k**$_\infty$**/k** e a onda é refratada assim como a velocidade sofre diminuição. De fato, pode-se aplicar a lei de Snell Descartes (USP-Educar, 2006) para calcular o ângulo que a frente de onda faz ao partir de uma profundidade infinita e entrar numa profundidade **h**.

$$\frac{\sin(\theta_h)}{\sin(\theta_\infty)} = \frac{c_h}{c_\infty}$$

A figura 18 mostra que quando a onda emitida por A' se desloca até B em um intervalo de tempo t, a onda emitida por A, neste mesmo intervalo de tempo, sofre um deslocamento menor até B', considerando que v2 < v1.

Sendo:A'B = v_1 t e AB' = v_2 t

Obtem-se:

$$\frac{A'B}{AB'} = \frac{v_1}{v_2}$$

Da geometria da figura 18, tem-se:

$$\sin\theta_1 = \frac{A'B}{AB}$$

$$\sin\theta_2 = \frac{AB'}{AB}$$

Dividindo as duas equações, obtém-se:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{A'B}{AB'} = \frac{v_1}{v_2}$$

Como $n_1 = v_c / v_1$ e $n_2 = v_c / v_2$, substituindo na equação anterior, obtem-se a expressão da lei de Snell Descartes:

$$\frac{\sin(\theta_1)}{\sin(\theta_2)} = \frac{n_2}{n_1}$$

O efeito de profundidade não pode ser calculado considerando-se apenas informações locais, o atraso da fase que é introduzido é acumulativo. Agora k é uma função de profundidade, que por sua vez é função de x_0. Assumindo $\Phi = 0$ para $x_0 = 0$, e que a constante de proporcionalidade $\lambda = 0$, a equação da fase agora será:

$$\Phi = -\omega t + \int_0^{x_0} k(x)dx$$

onde

$$k(x) = \frac{k_\infty}{\sqrt{\tanh(k_\infty h(x))}}$$

Desta forma será possível simular o efeito de refração da onda ao aproximar-se da costa e receber influência do fundo do mar.

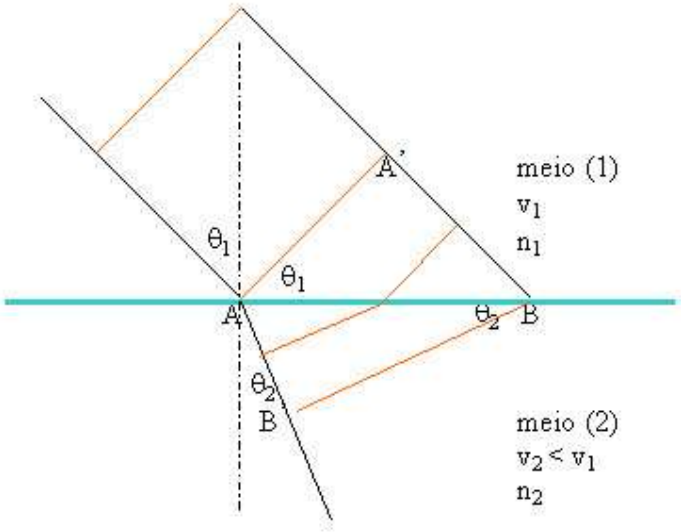


Figura 18 – Frente de onda na refração de acordo com a lei de Snell.

6.2.3.A onda se quebrando na costa

As teorias clássicas dizem que a medida que a onda se aproxima da costa, sua trajetória passa a ser elíptica ao invés da circular em águas profundas. Biesel (1952) propôs um modelo em que o eixo maior da elipse se alinhe com a inclinação do fundo do mar até que a profundidade se torne igual a zero (figura 19 e 20).

Figura 19 – Como a profundidade afeta a órbita

Figura 20 – Como a profundidade afeta a forma da onda

A elipse é uma curva plana, definida como o lugar geométrico dos pontos do plano para os quais a soma das distâncias a dois pontos fixos desse plano F1 e F2 é uma constante como ilustrado na figura 21.

O eixo S1S2 é denominado eixo maior da elipse e seu raio maior **s** é igual a metade do eixo maior. O eixo T1T2 é denominado eixo menor da elipse e seu raio menor t é igual a metade do eixo menor. A distância c é igual a distância do centro aos focos (F1 ou F2).

Os pontos F1 e F2 são denominados focos e a distância F1F2 é conhecida como distância focal da elipse. O quociente c/s é conhecido como excentricidade da elipse. Como, por definição, s > c, podemos afirmar que a excentricidade de uma elipse é um número positivo menor que a unidade.

Seja P(x, y) um ponto qualquer de uma elipse e sejam F1(c,0) e F2(c,0) os seus focos. Sendo 2s um valor constante com **c < s**, como vimos acima, podemos escrever (Macedo e Conci, 2005):

PF1 + PF2 =2s

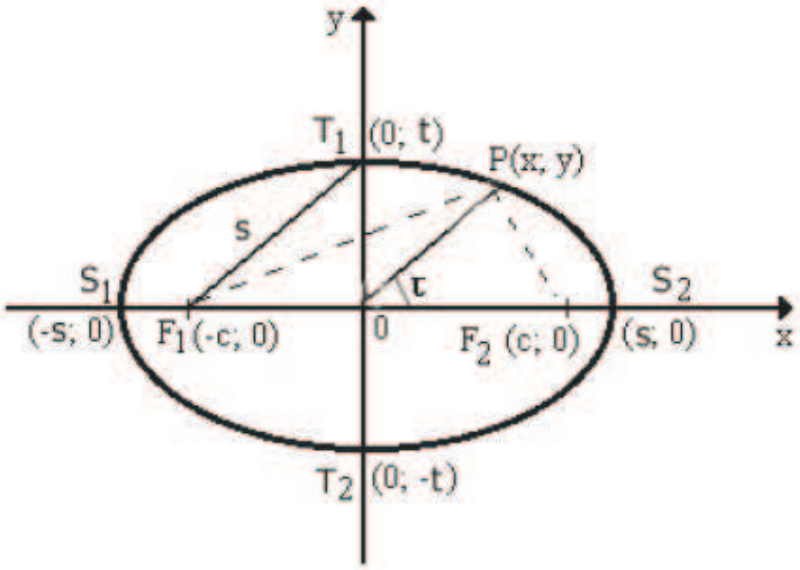


Figura 21 - Elipse

Uma elipse em coordenadas cartesianas é expressa da seguinte forma:

$$\frac{x^2}{s^2}+\frac{y^2}{t^2}=1,$$

para o caso do eixo maior estar no eixo dos x e

$$\frac{x^2}{t^2}+\frac{y^2}{s^2}=1,$$

para o caso do eixo maior estar no eixo dos y.

A equação polar da elipse em coordenadas polares é :

$$\rho^2=\frac{s^2t^2}{s^2\sin^2\tau+t^2\cos^2\tau},$$

onde s é a metade do eixo maior, *t* é a metade do eixo menor, ρ a distância entre o centro e um determinado ponto na borda da elipse, e τ. representa o ângulo que ρ faz com o eixo horizontal, sendo (ρ, τ) coordenadas polares da elipse (figura 23).

Para desenhar uma elipse com sua inclinação diferente de zero em relação ao eixo horizontal usa-se a matriz de rotação da figura 22.

$$\begin{vmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{vmatrix}$$

Figura 22 – Matriz de rotação

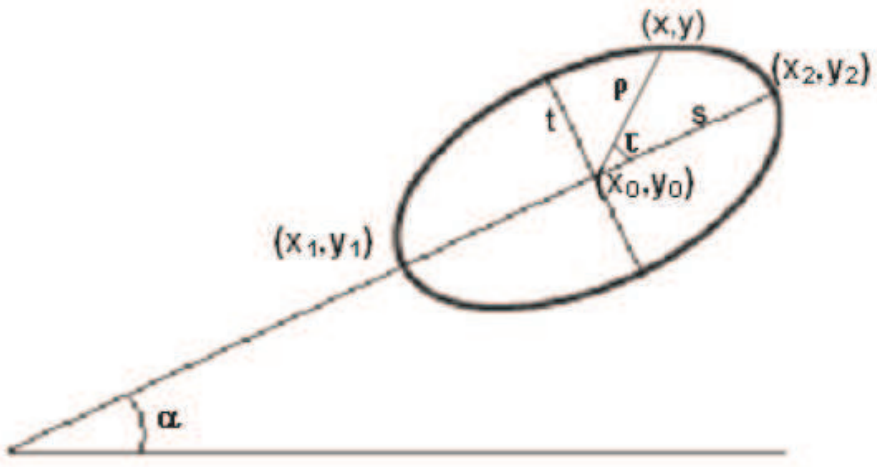


Figura 23 – Considerando a inclinação do fundo do mar na simulação

Cada coordenada (x,y) da elipse é transformada por essa matriz como mostra as equações abaixo.

$$x' = x\cos(\alpha) + y\sin(\alpha)$$
$$y' = -x\sin(\alpha) + y\cos(\alpha)$$

Um ponto da elipse pode ser determinado através de coordenadas polares da seguinte forma:

$$x = \rho\cos\tau$$

$$y = \rho\sin\tau\,,$$

como em uma elipse: [t ≤ρ ≤s], então:

$$x = s\cos\tau$$

$$y = t\sin\tau\,,$$

operando a rotação de um ponto juntamente com uma translação (x_0, y_0) temos:

$$x' = x_0 + s\cos\tau\cos\alpha + t\sin\tau\sin\alpha$$
$$y' = y_0 - s\cos\tau\sin\alpha + t\sin\tau\cos\alpha$$

Para adaptar a programação em GPU será utilizada a forma simplificada de Fournier-Reeves (1986) levando em consideração o custo computacional:

$$x = x_0 + r\cos\alpha.S_x.\sin\Phi + \sin\alpha.S_z.\cos\Phi$$
$$y = y_0 - r\cos\alpha.S_z.\cos\Phi + \sin\alpha.S_x.\sin\Phi \qquad (5)$$

Onde:

- **Φ** é a fase definida como no item 6.2.1;

- **senα = sen $\gamma e^{-k_0 h}$** , onde γ é a inclinação do fundo do mar em direção a trajetória da onda;

- **$S_x = 1/(1 - e^{-k_x h})$** é o incremento do eixo maior da elipse;

- **$S_y = S_x(1 - e^{-k_y h})$** é o decremento do eixo menor da elipse;

- **K_0** determina a influência da profundidade na inclinação na elipse;

- **K_x** é um fator de enlargamento do eixo maior da elipse;

- **K_y** é um fator de redução do eixo menor da elipse;

- **r** é o raio do disco.

É importante notar que $S_x \rightarrow \infty$ quando a profundidade $h \rightarrow 0$.

Com essa mudança da equação do movimento é possível obter um melhor realismo na forma da onda quando ela se quebra aproximando-se da costa e recebe a influência do fundo do mar, como mostra a figura 24.

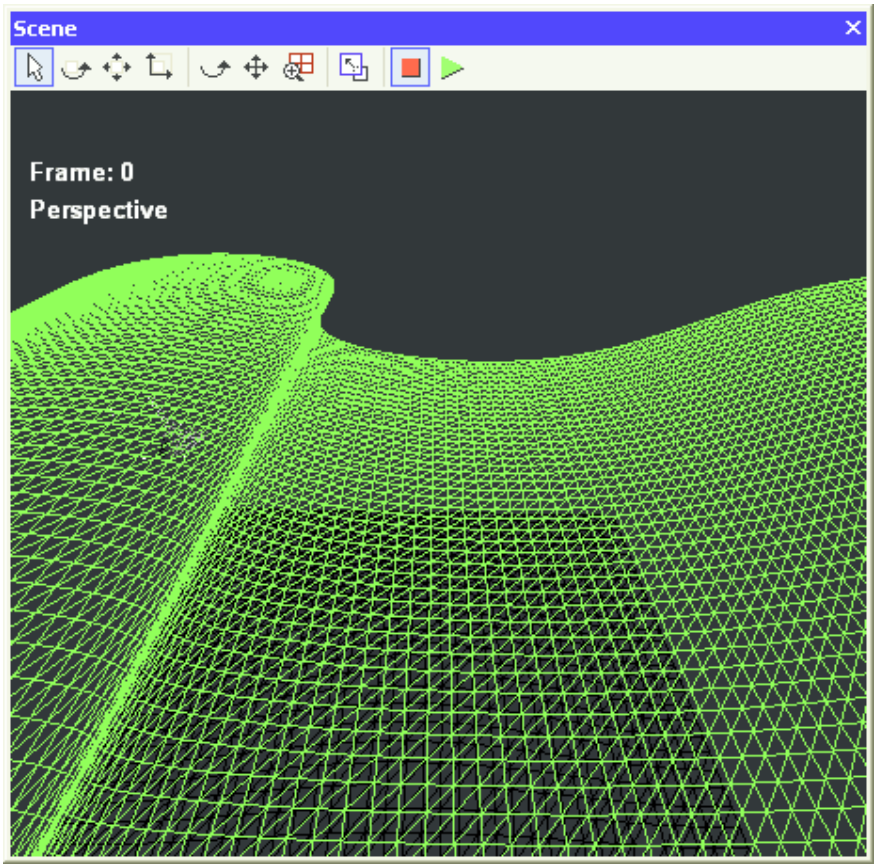


Figura 24 – Onda se quebrando na costa

6.2.3.1.Correção do modelo de Fournier-Reeves

Este modelo apresentado anteriormente no item 6.2.2 apresenta duas limitações. A primeira delas é que o modelo não permite que o fundo do mar tenha inclinações negativas, pois as ondas quebram-se na direção reversa da propagação da onda. Como pode ser visto na figura 25, esta limitação torna-se muito irreal. Já que uma possibilidade de fundo irregular com depressões e saliências é muito comum na natureza, especialmente em regiões rochosas ou em águas muito agitadas.

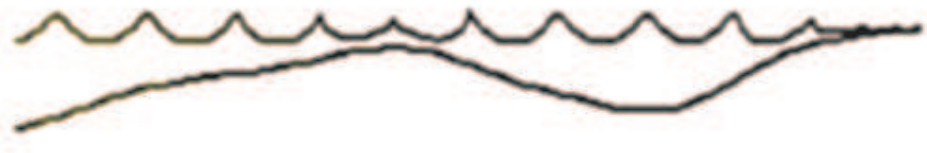


Figura 25 – Limitação do modelo de Fournier-Reeves.

Para resolver este problema usando o modelo de Fournier-Reeves, é necessário limitar a inclinação do eixo maior da elipse para um ângulo sempre positivo, considerado zero as inclinações negativas.

O segundo problema vem do fato que no modelo de Gerstner, os círculos descritos pelas partículas de água são restritos ao raio do disco isto é, **r <= 1/k**. Quando **r > 1/k** podem ocorrer laços na equação que representa a forma da onda que não ocorre na natureza como mostrado na figura 13, mas reapresentado na figura 26 desta vez mostrando a sua ocorrência também utilizando a trajetória elíptica na forma da equação 5. Neste caso, a solução pode ser fazer com que o eixo maior da elipse pode receber um valor maior que o disco de raio **r**. Esta ação simula o aparecimento da quebra da onda em profundidade média, mas ao aproximar-se da costa, o modelo torna-se inoperante pois são formados laços na forma da onda. Para este problema, a solução utilizada nesta dissertação foi limitar o eixo maior da elipse ao raio r do disco adquirindo a forma mostrada na figura 26.

Figura 26 – Correção de laços na formação da onda.

Para melhorar a aparência da geometria da queda da onda, aproximando-se das ondas em espiral (quebras mergulhando), Gonzato e Le Saec (1997) propuseram uma alteração na fase da equação de Fournier-Reeves que estica e torce a crista da onda como mostra a figura 27. Gonzato adicionou três novas funções chamadas *Strech, Orientation e Displacement*. A função *Strech* é usada para simular a aceleração da partícula na crista da onda. As funções *Orientation* e *Displacement* são combinadas para simular a influência da força da gravidade.

Figura 27 – Perfil da onda de Gonzato e Le Saec (1997)

6.2.3.2.A função *Strech*

Para manter a forma inicial da onda de Biesel, foi adicionado como parâmetro da função, um fator para esticar a crista da onda em direção ao eixo maior da elipse ao passo que nenhuma modificação é realizada na calha da onda. O parâmetro $St_{\max}$ define este tamanho máximo de enlargamento. Também é criado um parâmetro de escala K_s para determinar a influência da profundidade na função *Strech* .

Uma função parabólica chamada $Strech(\Phi, \lambda)$ de fase Φ é usada. O fundo da onda é obtido através do valor mínimo de y na equação paramétrica da elipse da seguinte forma:

$$\Phi_{\min} = a\tan\left(\frac{\sin(\alpha)S_x}{\cos(\alpha)S_y}\right)$$

E a função *Strech* é definida como:

$$Strech(\phi, St_{\max}) = \frac{1}{\pi^2}\left(St_{\max}\phi^2 - 2St_{\max}\Phi_{\min}\phi + St_{\max}\Phi_{\min}^2\right)$$

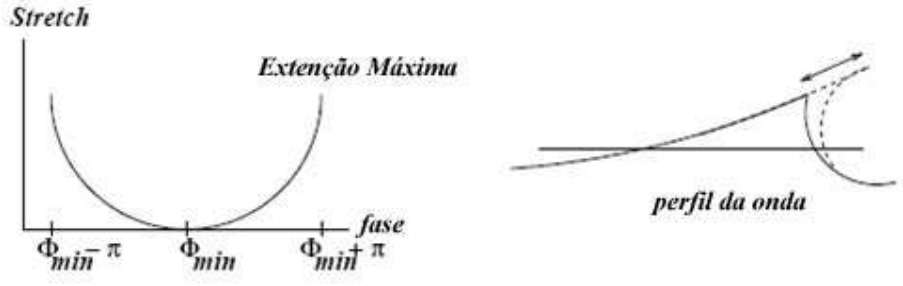


Figura 28 – Função Strech

6.2.3.3.Quebrando a onda

Em complemento a função Strech, é necessário alterar a forma da onda para representar a força da gravidade fazendo com que a mesma se quebre. Para isso, é criado uma função de orientação da crista da onda. Esta função altera a forma da

onda e progressivamente adiciona um novo estiramento que decrementa na direção do eixo maior da elipse até a posição vertical.

Esta função de orientação é combinada com uma função de deslocamento progressivo. Este valor é limitado a **2r** de modo que a não permitir a colisão da crista com o fundo da onda. Além disso, um fator de escala chamado K_d é utilizado para determinar a influência da profundidade na função de deslocamento.

No topo da crista da onda, a velocidade é importante, mas logo abaixo dela a velocidade é bem menor. Deste modo, esta função é dividida em três partes de forma empírica:

- $\Phi_{\min} - \pi \le \phi < \Phi_{\min} - \frac{\pi}{3}$ para a parte descendente;

- $\Phi_{\min} - \frac{\pi}{3} \le \phi < \Phi_{\min} + \frac{7\pi}{8}$ para a parte da calha;

- $\Phi_{\min} + \frac{7\pi}{8} \le \phi < \Phi_{\min} - \pi$ para a parte ascendente.

A função de deslocamento (chamada de $Displacement(\phi,2r)$) e a função de orientação (chamada de $Orientation(\phi,\beta)$) é definida como:

$$Orientation(\phi,\beta) = \begin{cases} \text{Se } (\Phi_{\min} - \pi \le \phi < \Phi_{\min} - \frac{\pi}{3}) \Rightarrow \text{ a equação} \\ \text{da linha passando por} \\ (\Phi_{\min} - \pi, -\frac{\pi}{4}) \,\&\, (\Phi_{\min} - \frac{\pi}{3}, \beta) \\ \text{Se } (\Phi_{\min} - \frac{\pi}{3} \le \phi < \Phi_{\min} + \frac{7\pi}{8}) \Rightarrow \beta \\ \text{Se } (\Phi_{\min} + \frac{7\pi}{8} \le \phi < \Phi_{\min} + \pi) \Rightarrow \text{ a equação} \\ \text{da linha passando por} \\ (\Phi_{\min} + \frac{7\pi}{8}, \beta) \,\&\, (\Phi_{\min} + \pi, -\frac{\pi}{4}) \end{cases}$$

$$Displacement(\phi,2r) = \begin{cases} \text{Se } (\Phi_{\min} - \pi \le \phi < \Phi_{\min} - \frac{\pi}{3}) \Rightarrow \text{ a equação} \\ \text{da linha passando por} \\ (\Phi_{\min} - \pi, 2r) \,\&\, (\Phi_{\min} - \frac{\pi}{3}, 0) \\ \text{Se } (\Phi_{\min} - \frac{\pi}{3} \le \phi < \Phi_{\min} + \frac{7\pi}{8}) \Rightarrow 0 \\ \text{Se } (\Phi_{\min} + \frac{7\pi}{8} \le \phi < \Phi_{\min} + \pi) \Rightarrow \text{ a equação} \\ \text{da linha passando por} \\ (\Phi_{\min} + \frac{7\pi}{8}, 0) \,\&\, (\Phi_{\min} + \pi, 2r) \end{cases}$$

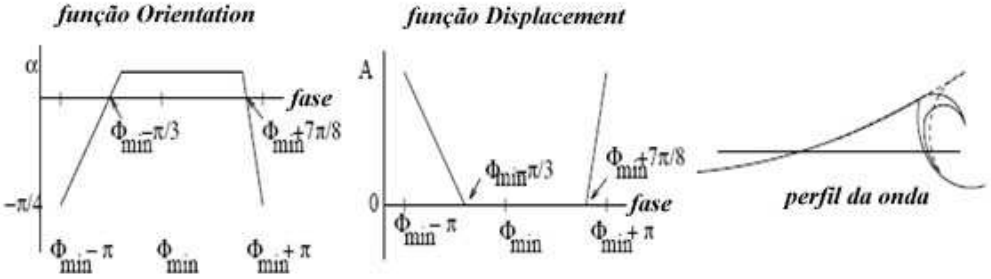


Figura 29 – Quebra da onda

Acrescentando a função *Strech*, a função *Displacement* e a função *Orientation*, a equação da onda se transforma em:

$$\begin{cases} x = x_0 + R\tau'_\beta S_x \sin(\Phi) + R\tau_\beta S_y \cos(\Phi) + Stretch(\Phi, St_{\max})\tau'_\beta e^{-K_s h} \\ + Displacement(\Phi,2r)\cos(Orientation(\Phi,\beta))e^{-K_d h} \\ y = y_0 - R\tau'_\beta S_y \cos(\Phi) + R\tau_\beta S_y \sin(\Phi) + Stretch(\Phi, St_{\max})\tau'_\beta e^{-K_s h} \\ + Displacement(\Phi,2r)\sin(Orientation(\Phi,\beta))e^{-K_d h} \end{cases}$$

Onde:

$\tau_\beta = \sin(\beta)e^{-0.1h}, \tau'_\beta = \sqrt{1-\tau_\beta}$, sendo β é a inclinação do fundo;

$S_x = \frac{1}{1-e^{-0.11h}}$, é o incremento do eixo maior;

$S_y = S_x(1-e^{-0.09h})$, é o decremento do eixo menor;

7
Computação óptica e renderização da superfície do mar

O capítulo anterior definiu a modelagem geométrica da onda, mas para que a cena fique realmente com realismo, é necessário cor. Como a água pode apresentar-se com uma aparência bem diferente dependendo do contexto da cena, é importante definir algumas categorias de efeitos de água. No caso deste trabalho, serão utilizadas técnicas para que a água assemelhe-se a aparência do oceano.

Este capítulo irá utilizar a programação de pixel na GPU com algumas técnicas aplicadas na renderização de oceanos. As técnicas aplicadas serão: reflexão por *environment mapping*, efeito Fresnel e HDR.

7.1.
Ondas capilares

Ao observar o mar, é possível verificar que além das grandes ondas que se formam, também existem pequenas ondulações ou ondas capilares na superfície das águas do oceano. Modelar a geometria de cada rugosidade destas levaria a um custo computacional muito grande e impraticável em cenas de tempo real. Para resolver este problema, será utilizada a técnica denominada ***bump mapping*** (Blinn, 1978).

Bump-mapping é uma técnica usada para adicionar realismo sem modificar a geometria do objeto (Conci e Azevedo, 2003). Essa técnica muda o tom do sombreamento nos *pixels*, produzindo uma ilusão de relevo no objeto renderizado. A cor de uma superfície está relacionada com o ângulo entre o vetor normal da superfície e a direção da luz. Em uma superfície plana, o vetor normal é o mesmo para toda a superfície, logo a cor da superfície será sempre a mesma. No *bump mapping*, as propriedades de refração da luz são usadas para simular a variação da luminosidade, cor e normal (figura 30). Para isso, a técnica consiste em perturbar

o vetor normal em vários pontos da superfície criando uma ilusão de que algumas partes da superfície estariam elevadas ou rebaixadas.

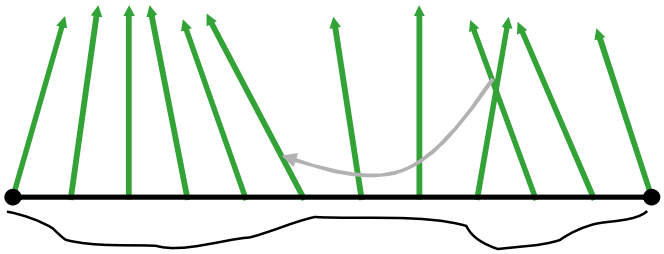


Figura 30 – Perturbação da normal da superfície

7.1.1. Mapa de normais

Para simular a rugosidade da superfície do oceano, será necessário utilizar um mapa de normais. Este mapeamento é semelhante a um mapeamento de textura 2D, mas aqui ao invés de cada ponto representar uma cor no padrão RGB, será armazenado no mapa um valor representando a normal em cada pixel. Isto é viável em tempo real devido ao uso da programação em GPU que possibilita operações rápidas de varreduras (*lookups*) em texturas que armazenam outros tipos de dados codificados como cores (Fernando e Kilgard, 2003). As coordenadas do vetor normal x, y e z são armazenadas como cores no padrão RGB. A coordenada x da normal é associada ao valor de R, a coordenada y é associada ao valor do canal G e a coordenada z ao valor de B. Normalmente a cor azul prevalece na textura de normais (figura 31), pois este corresponde à direção principal da normal ao oceano de acordo com o sistema de eixos aqui utilizados.

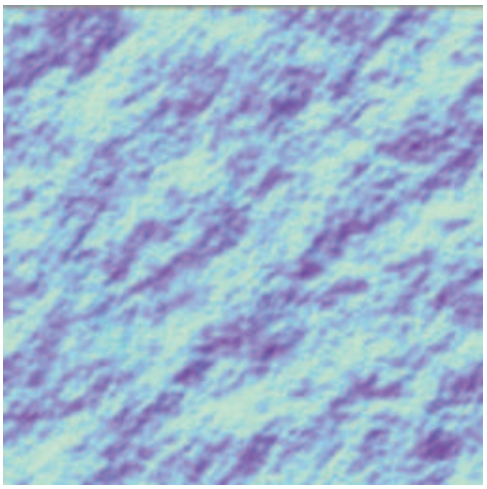


Figura 31 – Mapa de normais

7.1.2. Reflexão por bump mapping no espaço da tangente

Para que o *bump mapping* funcione corretamente para qualquer tipo de geométrica aplicada, é necessário realizar uma transformação. É necessário que o vetor de luz e *half vector* (Conci e Azevedo, 2003) compartilhem um sistema de coordenadas consistente com o vetor normal do mapa de normais. Ao invés de fazer com que todas as normais do mapa de normal se adequem ao espaço do objeto, é melhor e menos custoso transformar o vetor de luz para o sistema de coordenada do mapa de normais. Esse sistema de coordenadas é chamado espaço de textura, motivo pelo qual esta abordagem é chamada de *bump mapping* no espaço de textura ou no espaço da tangente (Fernando e Kilgard, 2003).

Para transformar estes vetores do espaço de objeto para o espaço da textura, será utilizada a seguinte matriz chamada TBN da figura 32:

$$\text{TBN} = \begin{vmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{vmatrix}$$

Figura 32 – Matriz (Tangente, Binormal e Normal)

O vetor binormal B e o vetor tangente T são as derivadas parciais em relação a s e t respectivamente. O vetor normal N é obtido com o produto vetorial entre o vetor binormal e a tangente (Everitt, 2001).

Optou-se por utilizar uma técnica chamada de *Reflective Tangent Space Bump Maping* (RTSBM). Esta técnica consiste em acessar o *cubemap* através da reflexão do vetor do observador (olho) com o mapa de normais que faz o *Bump Mapping* (Everitt, 2000).

Usando o RTSBM é necessário que o vetor do olho e a normal estejam no espaço do cubemap que normalmente é o espaço do mundo (world space) como mostra o diagrama da figura 33. Desta forma, acessa-se a textura de cubemap a partir do raio de reflexão como explicado no item 7.2.2.

Para transportar o vetor do olho para o espaço do mundo, multiplica-se a posição do vértice no espaço do objeto pela matriz "Model" disponível na linguagem Cg.

Para transportar o vetor normal que está no mapa de normais, é necessário multiplicar o vetor normal extraído do mapa de normais por uma matriz que o transporte para o espaço do mundo. Para isso, a seguinte matriz de rotação 3x3 será utilizada no *shader*:

```
tangentToWorldMatrix = objectToWorldMatrix * tangentToObjectMatrix
```

Onde a matriz tangentToObjectMatrix é obtida através da transposta da matriz TBN mostrada anteriormente (Bustamante e Celes, 2003).

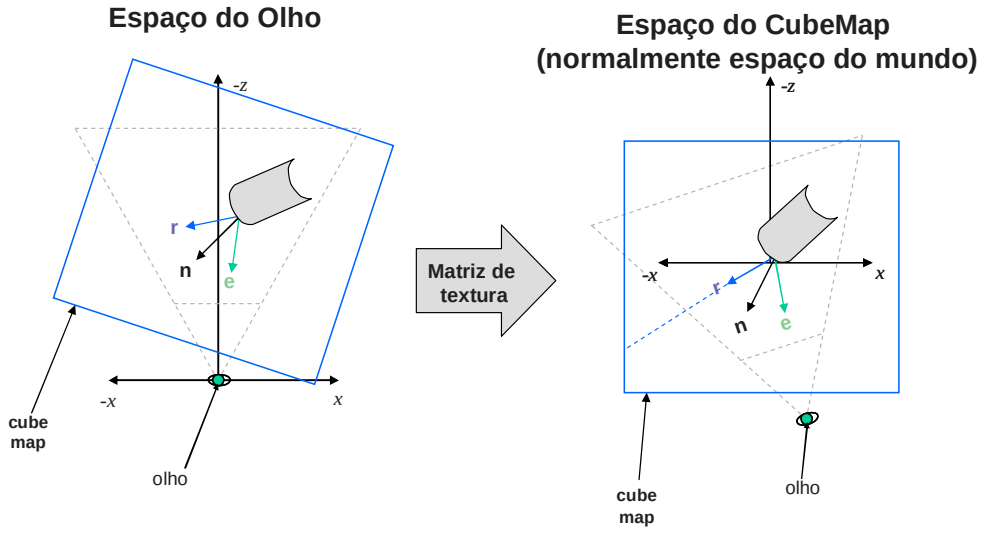


Figura 33 – Diagrama do mapeamento de reflexão

Nesta dissertação será utilizado o modelo de Phong (1975) para o cálculo de iluminação. Neste modelo a cor final é formada pela combinação das componentes: ambiente, difusa e especular.

7.2. Refletindo o ambiente

Para renderizar o oceano, será aplicada uma das propriedades físicas da água relacionado à óptica, a reflexão. Para tal, pode-se aplicar a técnica utilizada por (Vlachos, 2002), mas optou-se por utilizar a reflexão por bump mapping no espaço da tangente como descrito no item anterior. Após obter o vetor normal e o vetor do olho, é feito o mapeamento do ambiente ou *environment map*.

Environment mapping (EM), é um simples método, contudo poderoso para geração de aproximações de reflexão em superfícies curvas (Moller e Haines, 2002). A utilização do EM simula um objeto refletindo os seus arredores. Desta forma gera uma aparência cromática na superfície.

Para que o efeito funcione corretamente, o environment map parte do pressuposto que o objeto que receberá a reflexão estaria no mundo real a uma distância quase infinita das imagens criadas no ambiente a ser refletido. O refletor também não irá se refletir.

O objeto que receberá a reflexão fica envolvido dentro deste cubo. Traça-se um raio a partir do observador para o objeto e calcula-se o raio refletido nesta superfície até que ele toque um ponto em uma das faces do cubo. Este ponto dará a cor do ponto na origem do raio refletido, como mostra a figura 34.

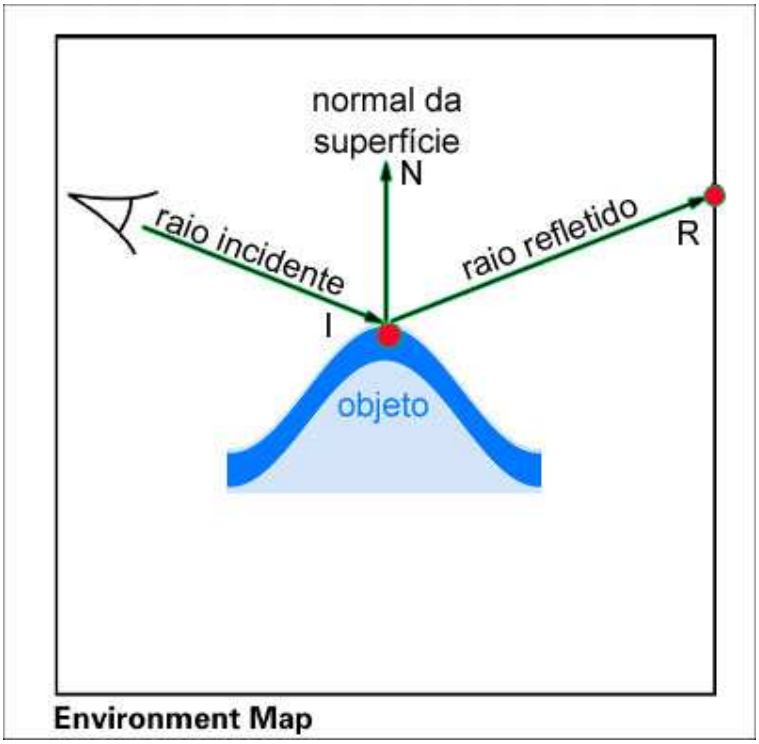


Figura 34 – Esquema de reflexão para formar o *environment map*

Para que a água não seja apresentada com uma aparência metálica (ou como um espelho perfeito), a reflexão será interpolada com a própria cor da água (azul) levando em consideração a profundidade da água (águas profundas mais escuras que águas rasas), ou seja, a reflexão será parcial e não total.

7.2.1.Mapa cúbico de ambiente (*Cubic environment map*)

Uma das formas mais utilizadas para usar o método de EM é utilizando a técnica de *cubic environment map*, ou *cube map* (Greene, 1986). O *cube map* é obtido colocando-se a câmera no centro do ambiente e projetando-se o ambiente nas faces de um cubo com seu centro posicionado no local da câmera. As imagens do cubo são utilizadas como o mapa do ambiente.

Para implementar esta reflexão utilizando a GPU, será utilizado um tipo de textura denominada *cube map texture*. Uma textura *cube map* é formado por seis imagens que se agrupam e se encaixam nas faces de um cubo o qual irá codificar o ambiente a ser refletido como mostra a figura 35.

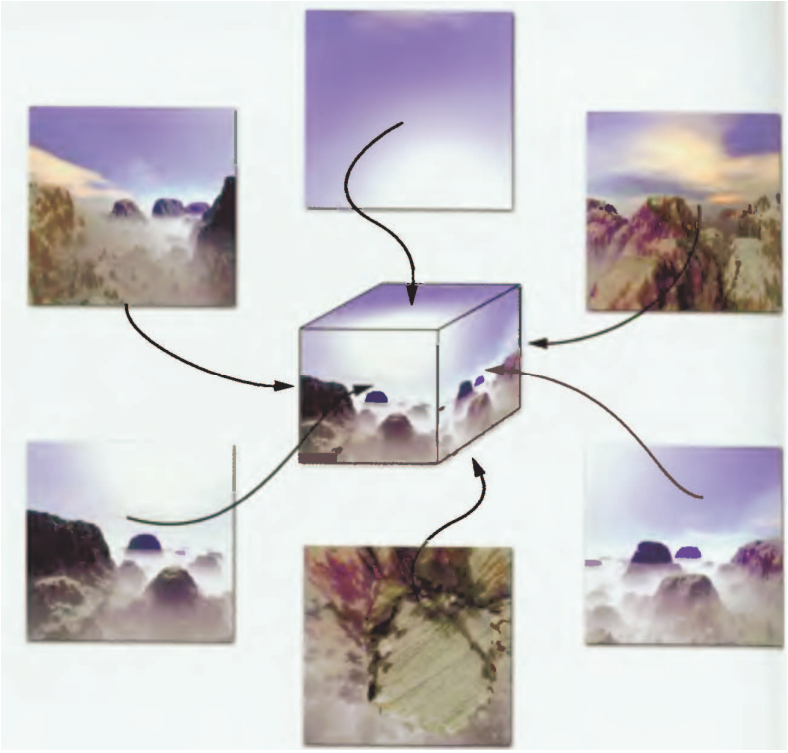


Figura 35 – Imagens de textura para um *cube map* (Fernando e Kilgard, 2003).

7.2.2.Calculando o raio de reflexão

Na figura 34, o vetor **I** (raio incidente) é originado da posição do observador e para a superfície do objeto. Quando I encontra a superfície, este é refletido na direção **R** baseado na normal da superfície **N**. Este segundo raio é o raio refletido.

O ângulo de incidência $\boldsymbol{\theta_I}$ é o mesmo ângulo de reflexão $\boldsymbol{\theta_R}$ para um refletor perfeito como um espelho. Pode-se representar o vetor de reflexão R da seguinte forma:

$$\mathbf{R = I - 2N(N.I)}$$

Como o cálculo de um vetor de reflexão é muito utilizado na computação gráfica, a linguagem Cg já possui uma função de reflexão na sua biblioteca padrão: `reflected (I, N)`. Onde I é o vetor representando o raio incidente e N o vetor representando a normal.

7.3. Reflexão de Fresnel

Quando a luz atravessa a interface entre dois materiais, por exemplo, ar e vidro, apenas uma parte da luz é transmitida dentro do novo material; outra parte

da luz se reflete na interface (Wloka, 2002). O nome para este efeito é **reflexão de Fresnel**. Em ângulos de incidência próximos a 90 graus, existe muita luz refletida e pouca luz refratada, o que explica a dificuldade de se olhar através da superfície água. A reflexão de Fresnel é mais visível em materiais semitransparentes tais como a água, o vidro, a pele, ou a pintura de um carro. A reflexão de Fresnel ocorre também ao se observar materiais opacos como o metal ou o papel.

A discussão da reflexão de Fresnel aqui se restringe aos limites de um único material, menos denso para o mais denso (ar-água).

7.3.1.A fórmula de Fresnel

A figura 36 descreve o cenário onde ocorre a reflexão de Fresnel se o índice do material **i** de refração **n** for menor que o índice do material **t** de refração **nt**.

O **θ** do ângulo varia de zero, quando o raio da luz é normal à superfície, à **θ = π/2**, quando o raio da luz é incidente à superfície.

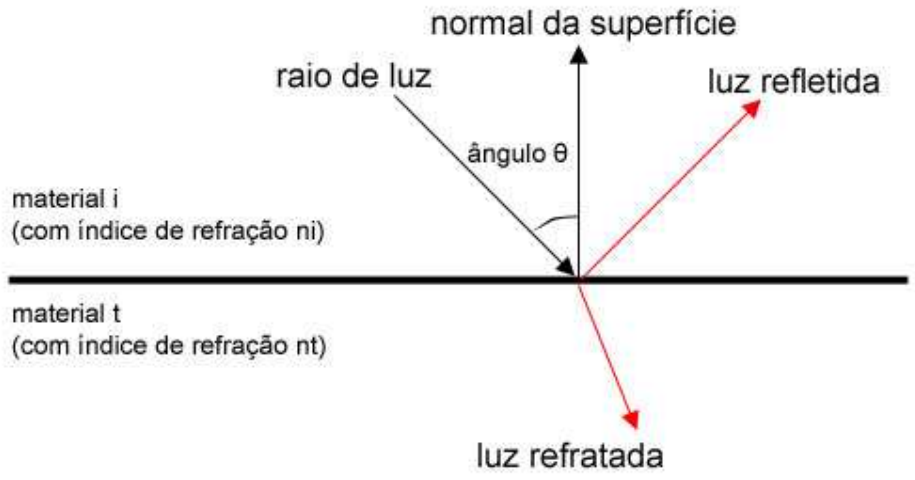


Figura 36 - Raio de luz viajando através de dois meios materiais.

A fórmula de Fresnel descreve o quanto de luz é refletida na interface do material e o quanto é refratada. A quantidade de reflexão depende do ângulo de incidência θ, da polarização da luz, da relação dos índices de refração $\mathbf{n_t/n_i}$, e do comprimento de onda da luz (uma vez que o índice de refração depende do comprimento de onda). A fórmula

$$\mathbf{R}(\theta) = \tfrac{1}{2}\,(\mathbf{R}^{\perp}(\theta) + \mathbf{R}_{\parallel}(\theta)) \qquad \textbf{(6)}$$

Onde $R^{\perp}$ e $R\|$ são respectivamente, a refletância para uma luz perpendicular ao plano de incidência e paralelo a ele representado na figura 37.

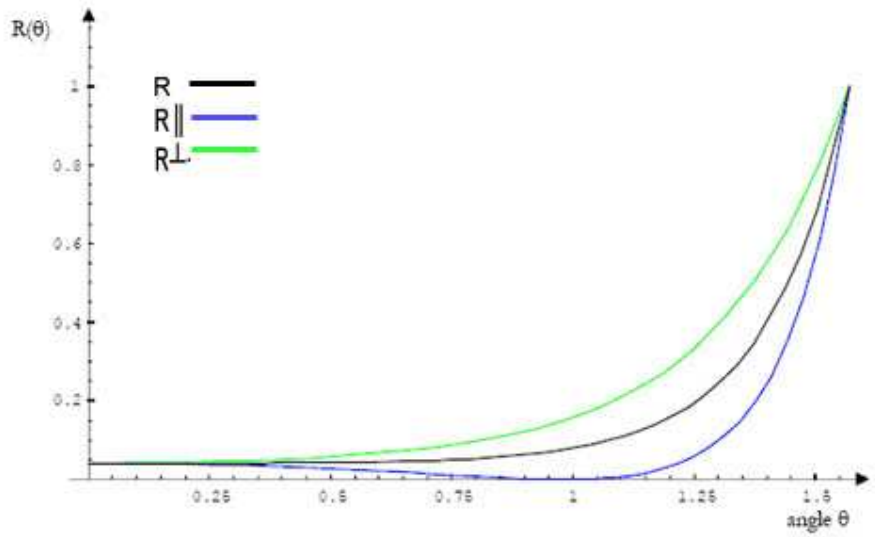


Figura 37 - O gráfico para R, $R^{\perp}$ e $R\|$

Assumindo-se que todas as luzes não são polarizadas e possuem o mesmo comprimento de onda, Wloka (2002) simplificou a equação 6 para reflexão de Fresnel de forma a diminuir o número de instruções a serem processadas pela GPU. A fórmula é:

$$R(\theta) \approx R_a(\theta) = R(0) + (1 - R(0))(1 - \cos(\theta))^5$$

Onde $R(0) = \dfrac{(n_1 - n_2)^2}{(n_1 + n_2)^2}$

Como estamos falando de reflexão na interface ar-água, substituindo-se os valores respectivamente de n_1 e n_2 para 1,000293 e 1,3333, obtém-se:

R(0) = 0,02037

A figura 38 mostra a aplicação da fórmula de Fresnel.

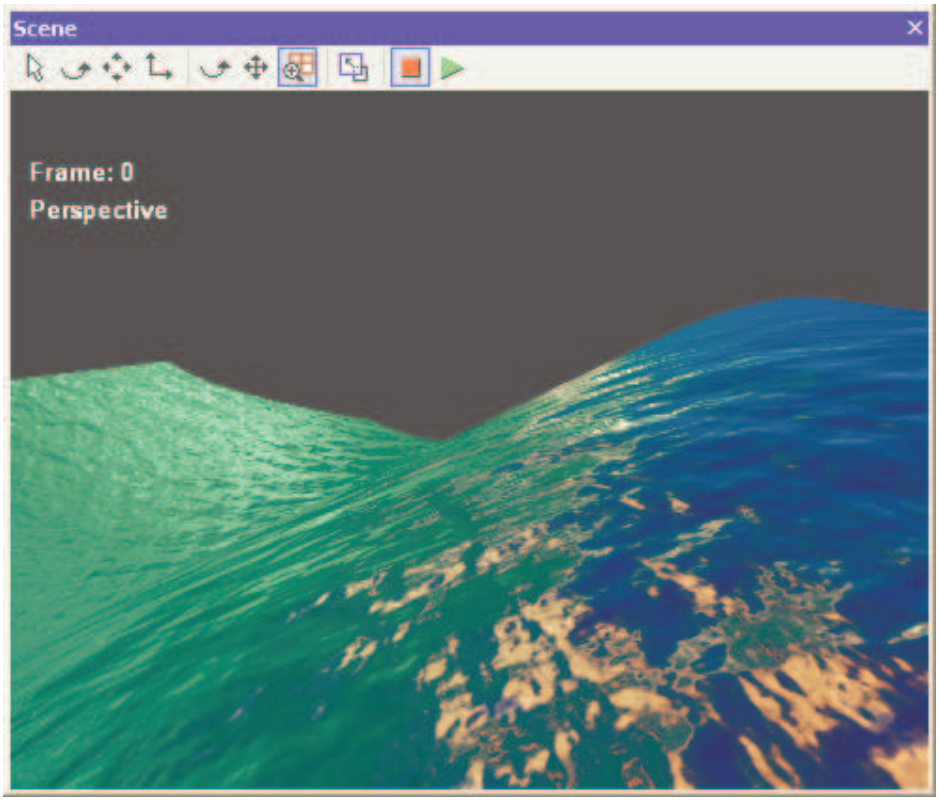


Figura 38 – Aplicando a reflexão de Fresnel na onda

O trecho de código a seguir mostra a implementação da reflexão de Fresnel explicado neste item:

```
...

float R0 = 0.02037;

//fresnelBias

fresnelPower = 5.0;

half facing = 1.0 - max(dot(E, Nw), 0);

half fresnel=R0+(1.0-R0)*pow(facing, fresnelPower);

...
```

7.4. HDR

HDR (High Dynamic Range) é a ciência do reconhecimento de diferentes níveis de intensidade da luz (St-Laurent, 2004). As imagem HDR (HDRI) são,

portanto, imagens que guardam não apenas informação de cor, mas também informação de intensidade de luz.

Ao contrário das imagens HDR, as imagens jpg, bmp, gif, guardam apenas informação de cor sem diferenciar as áreas com maior ou menor luminosidade. Existem varias discussões sobre a utilização de 8 bits de resolução por componente de cor seja o suficiente para representar todas as cores que o olho humano pode ver. Embora 256 tons por componente de cor seja o suficiente para representar uma cor, a intensidade da luz varia muito mais que isto, por exemplo, a luz emitida por uma vela e a luz emita pelo sol.

Utilizar a intensidade da luz corretamente, pode ser um forte fator para gerar mais realismo e confiabilidade no resultado da renderização de uma cena. Por isso, muitos estudos estão sendo feitos nesta linha no fenômeno chamado HDR.

7.4.1.Texturas de ponto flutuante

A razão principal da nova introdução de textura de ponto flutuante foi de habilitar suporte a aplicações que tenham como características a necessidade de uma faixa maior de valores e precisão do que atualmente é oferecido pelas texturas de 8 bits. O HDR é um exemplo desta necessidade de precisão.

7.4.2.Controle de exposição

O controle de exposição deve ser considerado com especial atenção ao implementar o HDR. Ambos, o olho humano e as câmeras devem estar preparados para se ajustar em diferentes condições de luz. Assim como a íris do olho humano se ajusta em ambientes com iluminações diferentes, a renderização de HDR precisa de um mecanismo similar para controlar a média de intensidade de iluminação na cena. Este mecanismo é chamado de **controle de exposição**.

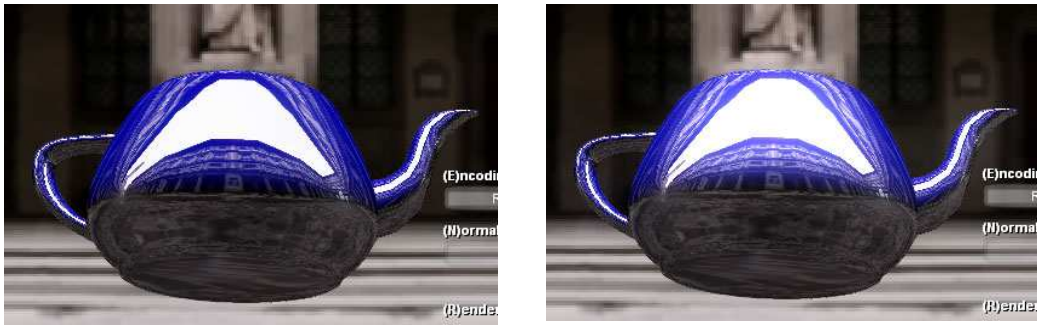


Figura 39 – HDR com baixa (esquerda) e média (direita) exposição

Um controle automático de exposição visa a ajustar o brilho médio da cena. O objetivo é de que o brilho médio fique em torne de 0,5 porque a escala de intensidade varia de zero a um. Se o brilho médio é conhecido, o controle de exposição pode ser calculado como (figura 39):

```
Exposição = 0,5 / brilho_médio
```

A exposição deve adaptar-se lentamente e não alterar instantaneamente. O seguinte trecho de código em Cg foi empregado no *shader* para o controle de exposição da cena do oceano:

```
Exposição = lerp( Exposição, 0,5 / brilho_médio, velocidade_de_ajuste_exposição)
```

Onde `lerp(min, max, passo)` é uma função de interpolação de valores entre o valor `min` e o valor `max` variando de `passo`.

7.4.3.Implementando o HDR na GPU

Nesta dissertação foi implementado o HDR utilizando-se das informações da textura *cube map*. O canal alfa da textura de *cube map* guarda a informação da intensidade de luz como mostra a figura 40. O algoritmo implementado consiste em multiplicar a informação do canal alfa pela componente de cor no ponto direcionado pela reflexão.

O trecho de código a seguir implementa o HDR com a utilização da textura de *cube map*:

```
reflection.rgb*=(1.0 + reflection.a*hdrMultiplier);
```

Onde `hdrMultiplier` é a variável para implementar controle de exposição.

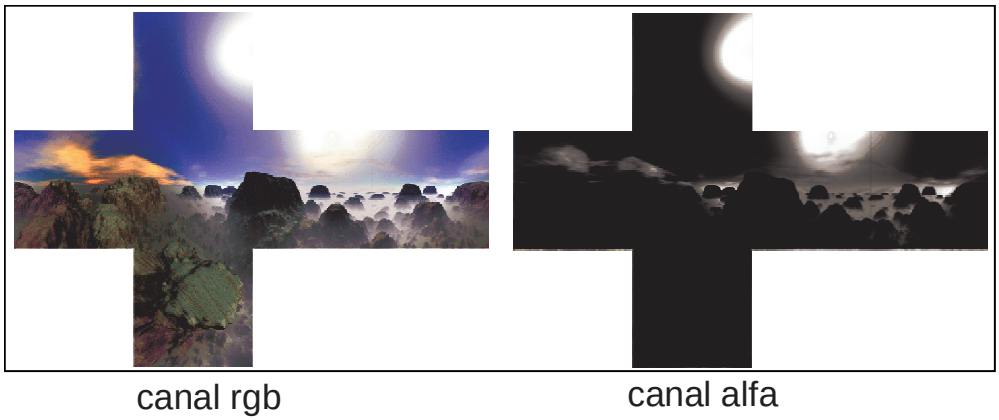


Figura 40 –Intensidade de luz no canal alfa da textura

8
Implementação e Resultados

O objetivo deste trabalho era criar uma simulação realística da superfície das ondas da superfície do mar. Para criar uma ferramenta produtiva e reutilizável, foi criado um *efeito* (arquivo.fx) de modo a simplificar a sua implementação tanto para um programador quanto para um artista 3D.

Utilizou-se o FX Composer 1.8 que é uma IDE para desenvolvimento de shader da NVIDIA® na linguagem Cg (Shader 2.0), e também foi testado aplicando-se o efeito dentro de um software de renderização 3D proprietário, o 3D MAX® da Discreet®. Todas as fotos apresentadas neste capítulo são retiradas da aplicação rodando em tempo-real.

Para renderizar a cena, foi utilizada as seguintes configurações de equipamento:

- Um computador AMD Athlon XP 1800 com 512 RAM;

- Uma placa de vídeo NVIDIA FX 5200 Shader 2.0.

O *shader* foi programado de forma que alguns parâmetros pudessem ser configurados pelo usuário. Estes parâmetros definem o comportamento visual e físico da onda do mar. De forma empírica e visual foram configurados valores para estes parâmetros de forma a atingir o objetivo final de uma cena realista.

Estes parâmetros que foram discutidos ao longo desta dissertação são exibidos na figura 41.

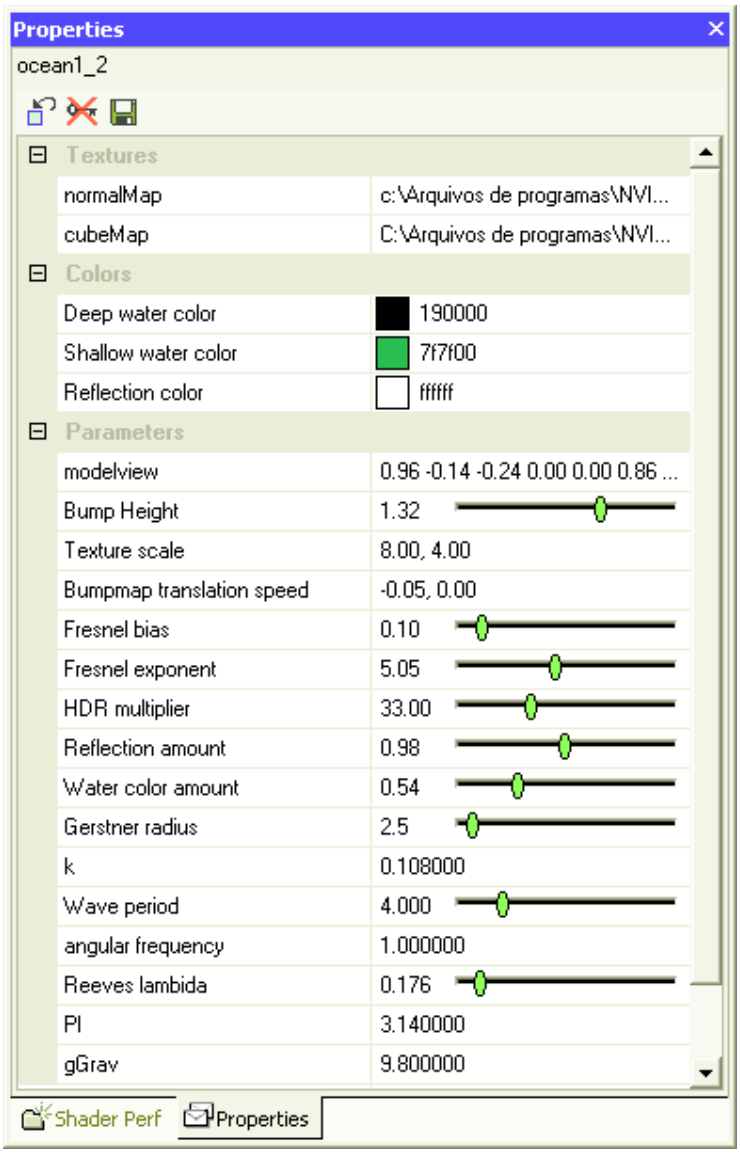


Figura 41 – Propriedades configuráveis do *shader*.

Manipulando-se essas propriedades, é possível:

- Alterar a granularidade das ondas capilares;

- Alterar a velocidade da partícula da água;

- Alterar a contribuição da reflexão de Fresnel e HDR no cálculo da iluminação;

- Criar ambientes noturnos, diurnos, sol, estrelas através do cubemap;

- Alterar o período, a freqüência e raio da onda;

- Contribuição do vento.

8.1.Definindo a informação de profundidade da água

Em várias equações desta dissertação são utilizadas a informação da profundidade (h) da superfície da água em relação a posição de repouso da partícula de água. Esta informação foi armazenada na coordenada y do modelo 3D que representa a água do oceano. Ou seja, a coordenada do vértice com um valor alto em y representa uma região profunda do mar. Um valor pequeno e próximo de zero na coordenada y do modelo 3D, representa a região de águas rasas do oceano.

8.2.Limitações na implementação do modelo proposto

Em função da utilização do Shader 2.0, houve algumas limitações no resultado. Estas limitações podem ser resolvidas com a migração para uma placa gráfica mais atual com suporte ao Shader 3.0 de forma a contribuir para trabalhos futuros desta dissertação.

Por exemplo, não foi possível obter o mapa de altura para definição da profundidade do terreno relativo ao fundo do mar, este recurso é suportado no Shader 3.0. Houve também uma limitação no número de instruções permitidas para o perfil do Shader 2.0. Atingiu-se o limite máximo de instruções e não foi possível implementar as rotinas de quebra de onda do item 6.2.3.3.

A seguir são mostradas as fotos das cenas obtidas com a implementação da solução proposta nesta dissertação.

8.3.Ondas de águas profundas com mar agitado

A figura 42 mostra uma seqüência da animação para uma onda de águas profundas. Com valores de configuração:

- Lâmbida = 0,176;
- Raio = 2,5;
- HDR = 33.

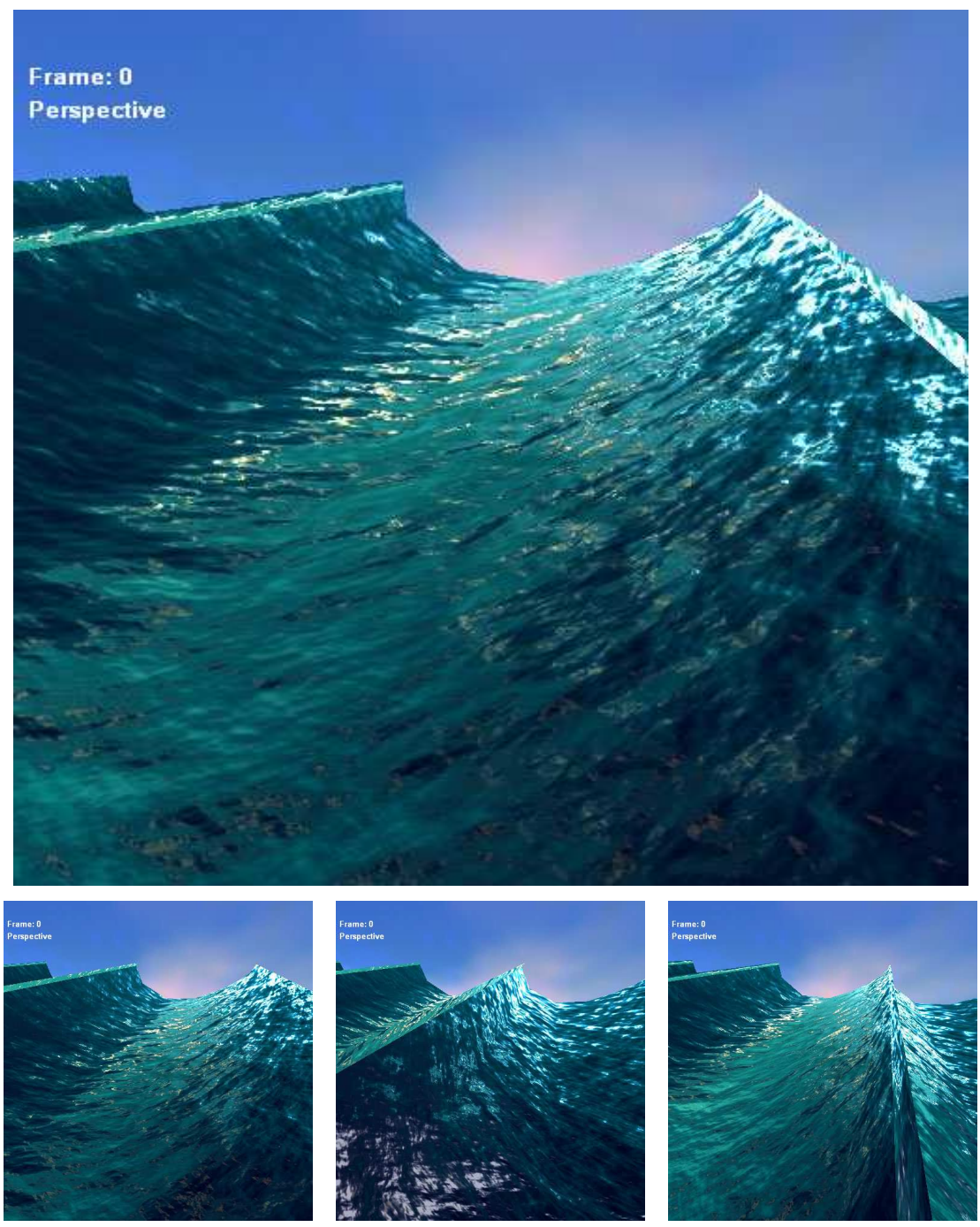


Figura 42 – Ondas em águas profundas

8.3.1.Ondas de águas profundas em ambiente noturno

A figura 43 mostra a figura com ondas de águas profundas. Os valores de configuração são os mesmos do item 8.1., mas agora com a representação do ambiente noturno.

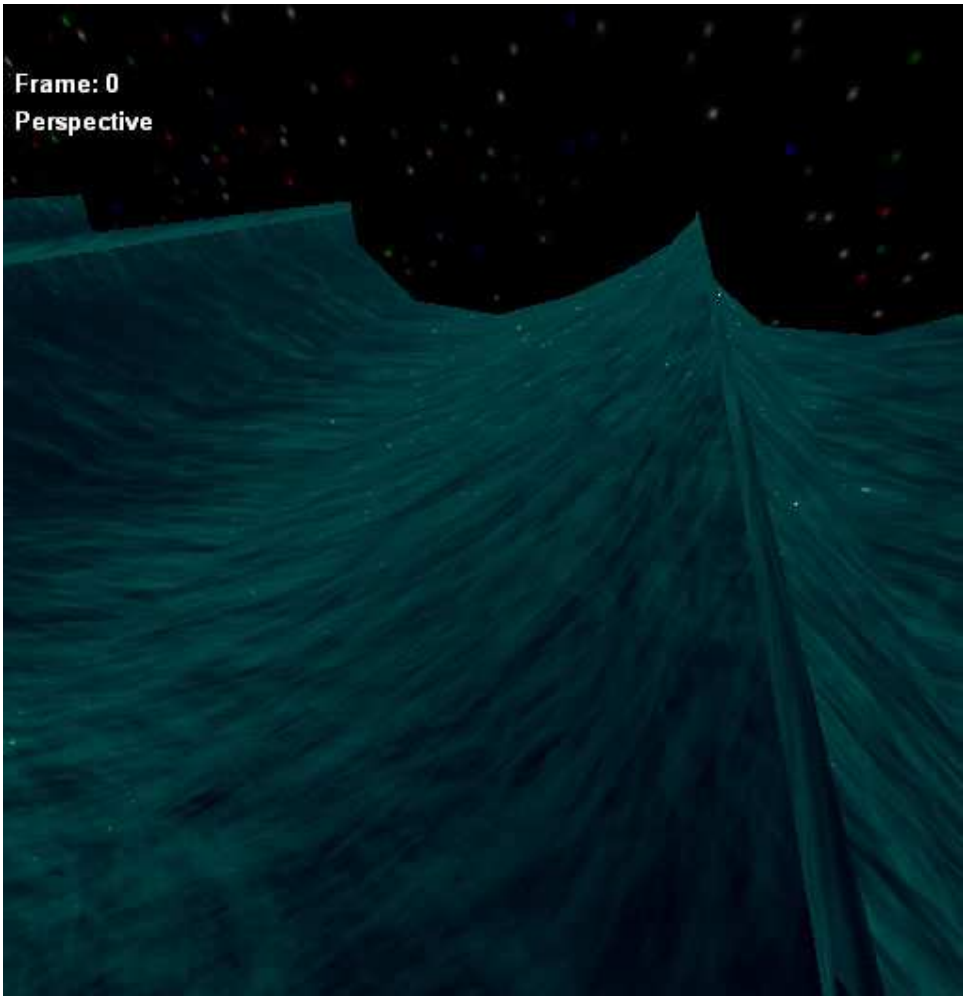
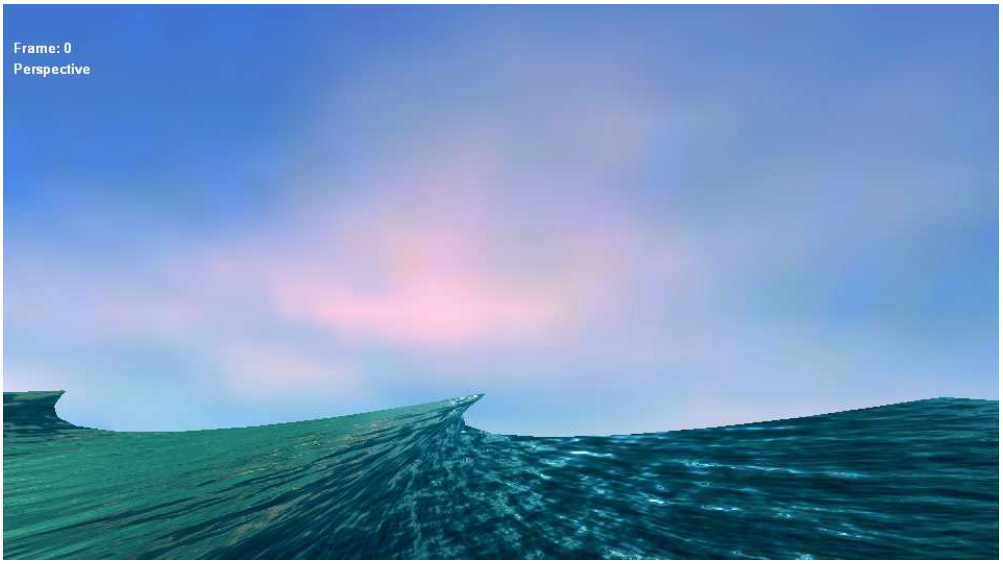


Figura 43 – Ondas de águas profundas em ambiente noturno.

8.3.2.Ondas de águas rasas

A figura 44 mostra a formação de ondas em águas rasas.



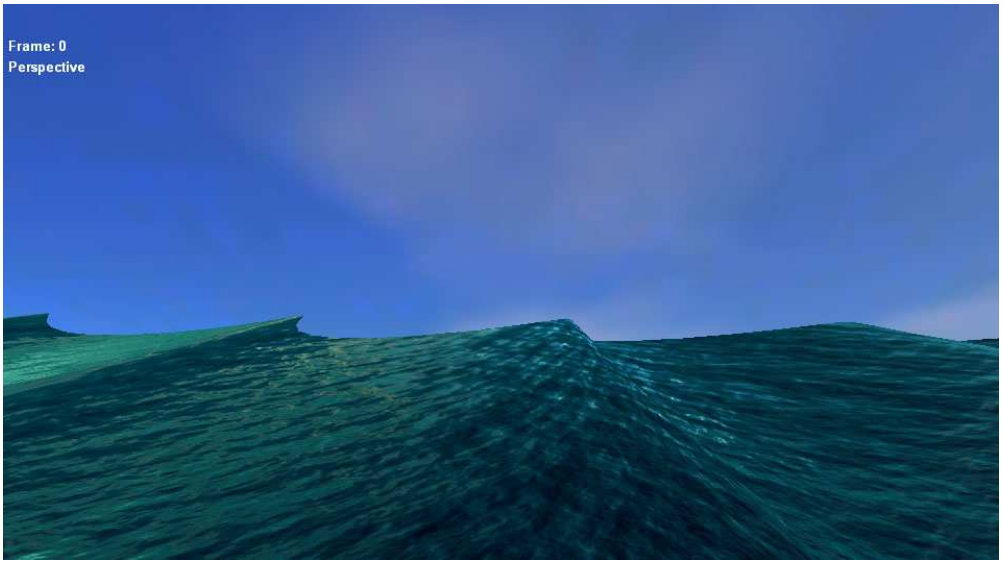


Figura 44 – Ondas de águas rasas

8.3.3.Águas profundas mar calmo

A figura 45 mostra a formação do mar em águas profundas sem ondas.

Figura 45 – Águas profundas sem ondas

8.3.4.Visão aérea

A figura 46 mostra uma visão aérea com *zoom* e outra mais distante de forma a exibir a refração da onda.

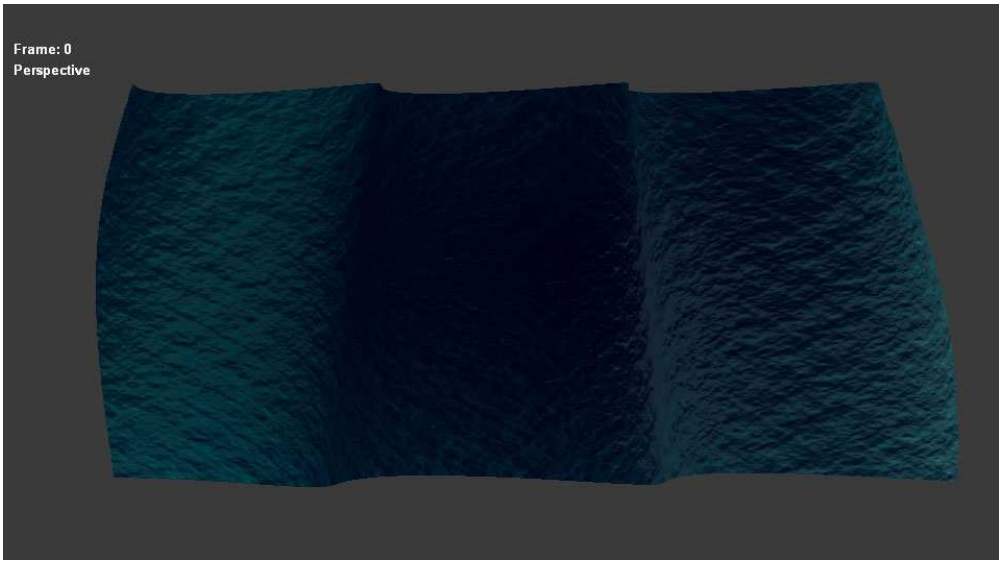


Figura 46 – Visão aérea

9
Conclusão

Foi proposto um sistema de rendering em tempo real para simulação de ondas oceânicas. Para alcançar um nível de renderização que caracterizasse uma aplicação em tempo real, todo o processamento e programação desta dissertação foi direcionado para a arquitetura e o hardware gráfico da GPU.

Foi abordado o comportamento das ondas em águas profundas, águas rasas com a influência do fundo do mar sobre a forma dessas ondas. Foi possível simular a modelagem geométrica baseada nas leis físicas e algumas configurações empíricas com atenção para as ondas se quebrando na costa oceânica. Simulou-se também a refração das ondas quando esta sente o fundo do mar e perde velocidade de fase de acordo com a lei de Snell Descartes, alterando as propriedades da onda tais como o comprimento de onda e velocidade, a medida que se aproxima da costa.

A renderização foi obtida com cálculo da iluminação baseada em sombreamento com relexão por bump-mapping no espaço da tangente, environment map, reflexão de fresnel e HDR tornando a visualização bem realista.

Os resultados obtidos como mostrado no capítulo 8 foram bem satisfatórios e convincentes quanto a visualização da superfície de um oceano com formação de ondas. Trata-se de um início de pesquisa, sendo assim, algumas alterações podem ser incluídas de forma a melhorar este trabalho e o realismo da cena.

Para aumentar o desempenho da renderização, pode-se acrescentar algoritmos de níveis de detalhe (LOD) para que os polígonos distantes da visão possam ser renderizados com menos detalhes (Bustamante e Celes, 2002).

Para representar a topologia do fundo do mar que foi utilizado nas equações da onda, poderíamos utilizar um mapa de altura que armazenasse estas informações. Assim o programa de vértice poderia ler este mapa e interpretar os

valores tornando mais refinado a visualização. Entretanto, este recurso só é possível com o perfil do *Shader 3.0* **(Gerasimov, Fernando e Green, 2004)** utilizando o recurso de Vertex Texture.

Nesta dissertação também não foi gerado o efeito de espuma e bolhas na superfície das ondas. O que pode ser gerado utilizando a física de sistema de partículas e autômatos celulares.

Pode-se também gerar o fenômeno de *caustics* que ocorre na superfície da água (Fernando, 2004).

A crista da onda ficou com uma aparência relativamente regular e reta que não existe na natureza da onda. Seria interessante criar irregularidades e ruídos na formação dessas cristas gerando padrões aleatórios utilizando a função *noise* (Perlin, 1985).

10
Bibliografia

(Adabala e Manobar, 2002) N. Adabala, S. Manohar, *Techniques for realistic visualization of fluids: a survey*, Comput. Graph. Forum volume 21 (1), pp. 65-81, 2002.

(Biesel, 1952) F. Biesel, *Study of wave propagation in water of gradually varying Depth*, Gravity Waves, pp. 243-253, U.S. National Bureau of Standards Circular 521, 1952.

(Blinn, 1978) J.F. Blinn, *Simulation of wrinkled surfaces*, Proceedings of SIGGRAPH'78, Comput. Graph, volume 12 (3), pp. 286–292. 1978.

(Bustamante e Celes, 2002) L. G. Bustamante, W. Celes, *Simulação e Visualização de Águas Oceânicas*, PUC-Rio, 2002.

(Bustamante e Celes, 2003) L. G. Bustamante, W. Celes, *Curso de Rendering em tempo real*, PUC-Rio, http://www.tecgraf.puc-rio.br/~gustavo/rendering/, 2003.

(Conci e Azevedo, 2003) E. Azevedo e A. Conci, *Computação Gráfica: Teoria e Prática*, Elsevier, 2003.

(Chen e Lobo, 1995) J. Chen, N. Lobo, *Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations*, Graphical models and Image Processing, pp.107-116, (março 1995).

(Chen, Lobo, Hughes e Moshell, 1997) J.X. Chen, N.V. Lobo, C.E. Hughes, J.M. Moshell, *Real-time fluid simulation in a dynamic virtual environment*, IEEE Comput.Graph. Appl., pp.52–61, (maio-junho 1997).

(Clua, 1999) E.W.G.Clua, *Modelagem Procedimental de Elementos da Natureza*, Dissertação de Mestrado, PUC-Rio, 1999.

(Everitt, 2000) C. Everitt, *Reflective Bump Mapping Lighting*, http://developer.nvidia.com/object/reflective_bump_mapping.html, 2002.

(Everitt, 2001) C. Everitt, *Mathematics of Per-Pixel Lighting*, http://developer.nvidia.com/object/mathematicsofperpixellighting.html, 2001.

(Fedkiw, Stam e Jensen, 2001) R .Fedkiw, J. Stam., H. W. Jensen. *Visual Simulation of Smoke,* Proceedings of SIGGRAPH 2001, pp. 15-22, 2001.

(Ferwwerda, 1999) J.A. Ferwerda, *Three varieties of realism in computer graphics*, Cornell Workshop on Rendering, Perception and Measurement, 1999.

(Fernando e Kilgard, 2003) R. Fernando, M. J. Kilgard. *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics,* Addison-Wesley Pub, 2003.

(Fernando, 2004) R. Fernando. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Addison-Wesley Pub, 2004.

(Fishman e Schachter, 1980) B. Fishman, B. Schachter, *Computer display of height fields*, Comput. Graph. 5 pp. 53–60, 1980.

(Fournier e Reeves, 1986) A. Fournier, W. T. Reeves, *A simple model of ocean waves,* SIGGRAPH´86, volume 20, pp. 75-84, 1986.

(Foster e Metaxas, 1996) N. Foster, D. Metaxas, *Realistic Animation of Liquids*, Graphical Models and Image Processing, 58(5), pp. 471-483, 1996.

(Foster e Metaxas, 1997a) N. Foster,D. Metaxas, *Modeling the motion of a hot, turbulent gas*, Proceedings of SIGGRAPH'97, pp. 181–188, 1997.

(Foster e Metaxas, 1997b) N. Foster, D. Metaxas, *Controlling fluid animation*, Proceedings of Computer Graphics International CGI'97, IEEE Computer Society Press, Menlo Park, CA, pp. 178–188, 1997.

(Foster e Metaxas, 2000) N. Foster, D. Metaxas, *Modeling water for computer animation*, Commun. ACM volume 43 (7), pp. 60–67, 2000.

(Foster e Fedkiw, 2001) N. Foster, R. Fedkiw, *Practical Animation of Liquids,* Proceedings of SIGGRAPH 2001, pp. 23-30, 2001.

(Fox e Mitchell, 1984) D. Fox, M. Waiter, *Carla´s Island film*, Computer Animation Primer, 1984.

(Garrison, 2004) Garrison, *An Invitation to Marine Science*, 4th Ed. Capítulo 10, pp. 237-258., Disponível em

http://www4.ncsu.edu/eos/users/c/ceknowle/public/chapter10/index.html#Discussion, (20/10/2006).

(Geiger, Leo, Ramussen, Losasso e Fedkiw, 2006) Geiger, W., Leo, M., Rasmussen, N., Losasso, F. and Fedkiw, R., *So Real It'll Make You Wet*, SIGGRAPH 2006 Sketches and Applications, 2006.

(Gerasimov, Fernando e Green, 2004) P. Gerasimov, R. Fernando, S. Green, *Shader Model 3.0: Using Vertex Texture*, NVidia Corporation, 2004.

(Gerstner, 1809) F.J. Gerstner, *Theorie der wellen*, Ann. der Physik 32, pp. 412-440, 1809.

(Gonzato, Le Saec, 1997) J.C. Gonzato, B. Le Saec, *A phenomenological model of coastal scenes based on physical considerations*, 8th Eurographics Workshop on Computer Animation and Simulation, pp. 137-148, 1997.

(Gonzato, Le Saec, 1999) J.C. Gonzato, B. Le Saec, *On Modeling and Rendering Ocean Scenes (Diffraction,Surface Tracking and Illumination)*, Proc. of Winter School on Computer Graphics'99, pp.93-101, 1999

(Gonzato, Le Saec, 2000) J.C. Gonzato, B. Le Saec, *On Modeling and Rendering Ocean Scenes*, Journal of Visualisation and Computer Simulation, 11, pp. 27-37, 2000.

(Goss, 1990) M.E. Goss, A *real-time particle system for display of ship wakes*, IEEE Comput. Graph. Appl., volume 10 (3), pp. 30-35, 1990.

(Greene, 1986) N. Greene, *Environment Mapping and Other Applications of World Projections*, IEEE Computer Graphics and Applications, vol 6 (11), pp. 21-29. (novembro 1986).

(Hurley, 2001) K. Hurley, *Lighting Techniques for Games*, Xtreme Game Developers Conference, 2001.

(Irving, Guendelman, Losasso e Fedkiw, 2006) Irving, G., Guendelman, E., Losasso, F. and Fedkiw, R., *Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques*, SIGGRAPH 2006, ACM TOG 25, pp. 805-811, 2006.

(Imamiya e Zhang, 1995) A. Imamiya, D. Zhang, *Modelling breaking ocean waves, influence of floor and refraction*, Pacific Graphics 95, 1995.

(Iglesias, 2004) A. Iglesias, *Computer graphics for water modeling and rendering: a survey*, Future Generation Computer System, volume 20 (8), pp. 1355-1374, (novembro 2004).

(Kass e Miller, 1990) M. Kass, G. Miller, *Rapid, stable fluid dynamics for computer graphics*, ACM Press, New York, NY, USA, 1990.

(Kinsman, 1965) B. Kinsman, *Wind Waves: Their Generation and Propagation on the Ocean Surface*, Prentice Hall, 1965.

(Losasso, Fedkiw e Osher, 2006) Losasso, F., Fedkiw, R. and Osher, *Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow*, Computers and Fluids 35, pp. 995-1010, 2006.

(Macedo e Conci, 2005) M. Macedo, A. Conci, *Uso da Transformada de Hough na Vetorização de Moldes e Outras Aplicações*, 2005.

(Max, 1981) N.L. Max, *Vectorized procedural models for natural terrain:waves and islands in the sunset*, Proceedings of SIGGRAPH'81, Comput. Graph, Volume 15 (3) pp. 324, 1981.

(Miller e Pearce, 1989) G. Miller, A. Pearce, *Globular dynamics: a connected particle system for animating viscous fluids*, Comput. Graph. Volume 13 (3), pp. 305–309, 1989.

(MSDN, 2003) MSDN Library, *HLSL Shaders*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/HLSL_Shaders.asp (20/10/2006), 2003.

(Moller e Haines, 2002) T. Moller, E. Haines, *Real-time Rendering (second edition)*, A K Peters, 2002.

(NVIDIA, 2002) NVidia Corporation, *User Guide: CgFX Plug-in for 3ds max*, 2002.

(NVIDIA, 2003) NVidia Corporation, *Cg Toolkit 1.5*, Disponível em http://developer.nvidia.com/object/cg_toolkit.html (20/10/2006), 2003.

(OceanWeather, 2006) OceanWeather, *Oceanweather Inc.*, Disponível em http://www.oceanweather.com/data/, (01/05/2006).

(Premoze e Ashikhmin, 2001) S. Premoze, M. Ashikhmin , *Rendering Natural Waters*, Eighth Pacific Conference on Computer Graphics and Applications, (Outubro 2000).

(Peachey, 1986) D. R. Peachey, *Modeling waves and surf*, SIGGRAPH´86, volume 20, pp. 65-70, 1986.

(Perlin, 1985) K. Perlin, *An image synthesizer*, Computer Graphics (SIGGRAPH ’85 Proceedings), volume 19, pp. 287-296, (Julho, 1985).

(Phong, 1975) Bui Tuong Phong, *Illumination for Computer Generated Images*, Comm. ACM, Vol 18(6), pp.311-317, June 1975.

(Pyramid, 1981) Pyramid, *Pyramid Catalogue: Pyramid*, Box 1048, Santa Monica, 1981.

(Rost, 2004) R. J. Rost. *OpenGL® Shading Language*, Addison-Wesley Pub, 2004.

(Reeves, 1983) W.T. Reeves, *Particle systems—a technique for modeling a class of fuzzy objects,*: Proceedings of SIGGRAPH’83, Comput. Graph. Volume 17 (3) pp. 359–376, 1983.

(Sims, 1990) K. Sims, *Particle animation and rendering using data parallel computation*, Proceedings of SIGGRAPH’90, Comput.Graph. volume 24 (4), pp. 405–413, 1990.

(St-Laurent, 2004) S. St-Laurent, *Shaders for Game Programmers and Artists*, Premier Press, 2004.

(Ts´o e Barsky, 1987) P. Y. Ts’o, B. A. Barsky, *Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping,*. ACM Transactions on Graphics, volume 6 pp. 191-214, (Julho 1987).

(Tomczak, 2002) M. Tomczak, *An Introduction to Physical Oceanography*, Disponível em http://www.es.flinders.edu.au/~mattom/IntroOc/index.html, (20/08/2006).

(Tonnesen, 1991) D. Tonnesen, *Modeling liquids and solids using thermal particles,*: Proceedings of Graphics Interface'91, pp. 255–262, 1991.

(Tessendorf, 1999) J. Tessendorf, *Simulating ocean water,* SIGGRAPH'99 Course Notes, 1999.

(USP-Educar, 2006) Programa Educar CDCC – USP SC , *Refração: Fundamentos teóricos*, http://educar.sc.usp.br/otica/refracao.htm#lei, (04/11/2006).

(Vlachos, 2002) A. Vlachos, J. Isidoro, C. Oat, *Rippling Reflective and Refractive Water*, ShaderX, Wolgang Engel. Wordware, 2002.

(Wiki, 2006) Wikipédia, *Wikipédia: Força de Coriolis*, Disponível em http://pt.wikipedia.org/wiki/For%C3%A7a_de_Coriolis, (20/10/2006).

(Witting, 1999) P. Witting, Computational fluid dynamics in a traditional animation environment, in: Proceedings of SIGGRAPH'99, pp. 129–136, 1999.

(Wloka, 2002) M. Wloka, *Technical report: Fresnel Reflexion,* NVIDIA Corporation, (Junho 2002).